PennState

Proceedings of

ICCM 2016

14th International Conference on Cognitive Modeling

August 3-6, 2016

The Pennsylvania State University, University Park, Pennsylvania

Edited by

David Reitter

Frank E. Ritter

Papers in this volume may be cited as:

Author, F. (2016). Title. In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Rev. 161216
ISBN 978-0-9985082-0-7

Conference Committees

General and Program Chairs

David Reitter	The Pennsylvania State University
Frank E. Ritter	The Pennsylvania State University

Program Committee

Eduardo Alonso	City University London
Erik Altmann	Michigan State University
Thomas Barkowsky	University of Bremen
Martin Baumann	University of Ulm
Roman Belavkin	Middlesex University
Thierry Bellet	IFSTTAR France
Jennifer Bittner	Air Force Research Laboratory
Jelmer Borst	University of Groningen
Mike Byrne	Rice University
Rick Cooper	University of London
Gary Cottrell	University of California, San Diego
Chris Dancy	Bucknell University
Vladislav Daniel Veksler	Air Force Research Laboratory
Adele Diederich	Jacobs University Bremen
Paula Droege	The Pennsylvania State University
Wai-Tat Fu	University of Illinois at Urbana-Champaign
Danilo Fum	University of Trieste
Francesco Gagliardi	
Moojan Ghafurian	The Pennsylvania State University
Kevin Gluck	Air Force Research Laboratory
Cleotilde Gonzalez	Carnegie Mellon University
Glenn Gunzelmann	Air Force Research Laboratory
Dan Guzek	The Pennsylvania State University
Christian Janssen	Utrecht University
Gary Jones	Nottingham Trent University
Ion Juvina	Wright State University
Mark Keane	University College Dublin
Matthew Kelly	Carleton University
Bill Kennedy	George Mason University
David Kieras	University of Michigan
Joseph Krems	Technische Universität Chemnitz
Bernd Kröger	University Hospital Aachen
Johan Kwisthout	Donders Institute for Brain, Cognition and Behaviour
John Laird	University of Michigan
Christian Lebiere	Carnegie Mellon University
Lyle Long	The Pennsylvania State University
Luís Macedo	University of Coimbra
Ralf Mayrhofer	University of Göttingen
Victor Middleton	Wright State University
Junya Morita	Shizuoka University
Shane Mueller	Michigan Technological University

Christopher Myers	Air Force Research Laboratory
Josef Nerb	University of Education, Freiburg
Hansjoerg Neth	University of Konstanz
David Noelle	University of California, Merced
Enkhbold Nyamsuren	The Open University
Alexander Ororbia	The Pennsylvania State University
David Peebles	University of Huddersfield
Marco Ragni	University of Freiburg
Nele Rußwinkel	Technische Universität Berlin
Dario Salvucci	Drexel University
Ute Schmid	University of Bamberg
Mike Schoelles	Rensselaer Polytechnic Institute
Lael Schooler	Syracuse University
Holger Schultheis	University of Bremen
Chris Schunn	University of Pittsburgh
Barry Silverman	University of Pennsylvania
Jennifer Spenader	University of Groningen
Robert St. Amant	North Carolina State University
Christopher Stevens	University of Groningen
Terrence Stewart	University of Waterloo
Andrea Stocco	University of Washington
Ron Sun	Rensselaer Polytechnic Institute
Niels Taatgen	University of Groningen
Julia Taylor	Purdue University
Manfred Thuering	Technische Universität Berlin
Greg Trafton	Naval Research Lab
Robert West	Carleton University
Sharon Wood	University of Sussex
Xiaolong Luke Zhang	The Pennsylvania State University
Changkun Zhao	IBM Watson
Iraide Zipitria	The University of the Basque Country
Hedderik van Rijn	University of Groningen
Iris van Rooij	Radboud University Nijmegen
Leendert van Maanen	University of Amsterdam
Marieke van Vugt	University of Groningen

Tutorial Committee

William G. Kennedy (Chair)	George Mason University
Ion Juvina	Wright State
Laura Hiatt	Naval Research Laboratory
Randolph M. Jones	Soar Technology
David Peebles	University of Huddersfield
Rob Thomson	US Military Academy, West Point
Nele Russwinkel	Technische Universität Berlin
Matt Walsh	Air Force Research Laboratory

Table of Contents

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Introduction to ICCM 2016

Frank E. Ritter and David Reitter

It is our pleasure to introduce the proceedings of the 14th International Conference on Cognitive Modeling. We started in Berlin where we were called the *First European Workshop on Cognitive Modeling*. This community had held earlier workshops on the Soar and ACT-R cognitive architectures held in Europe. With later conferences, we started to go back and forth across the Atlantic, and also setup a pattern of going to mainland Europe and England alternatively (where proceedings sometimes appeared as 'modelling', the British spelling).

14	Penn State	7	U. of Trieste, Italy
13	Groningen, Netherlands	6	Carnegie Mellon U.
12	Carleton U., Canada	5	Bamburg, Germany
11	TU Berlin, Germany	4	George Mason U.
10	Drexel U.	3	Groningen, Netherlands
9	U. of Manchester, England	2	Nottingham, U.K.
8	U. of Michigan	1	Berlin, Germany

We have been published by Erlbaum and several university presses. Currently, we are self-published on the web, with several repositories.

In this conference, we start with a useful and lively tutorial program, an aspect of most instances of this conference. These tutorials are designed to help experienced and inexperienced learn about new and also existing techniques, and also cover new architectures in detail, enough that potential users can learn about them and potentially adopt them.

In the past, this conference has preferred model + data + comparison. This year we were somewhat more broad in what was accepted. The proceedings have interesting papers for the presentations and posters on topics such as modeling tools, models of learning and memory, the use of models in other systems, models of adversarial systems, modeling natural language, technology for modeling, and modeling for math. We also have models on physiology and neural level models.

Out of 65 submissions, 33 papers were accepted as full papers (51 percent). We also include a number of additional abstracts in these proceedings.

The Allen Newell Award for the best student-led paper was given to Peter Duggins, Terrence C. Stewart, Xuan Choo, and Chris Eliasmith for their paper titled *Effects of Guanfacine and Phenylephrine on a Spiking Neuron Model of Working Memory*. We also note, as honorable mention, the papers by Hrvoje Stojic, Henrik Olsson, and Pantelis P. Analytis, by Peter Lindes and John E. Laird, by Dan Parker and Daniel Lantz, and by Ehsan Khosroshahi, Dario Salvucci, Bella Veksler, and Glenn Gunzelmann.

We would like to thank our sponsors, the US National Science Foundation (NSF BCS-1613241), the College of Information Sciences and Technology (Penn State), Charles River Analytics, Soar Technology, and Applied Cognitive Systems, LLC. These sponsors have helped make the conference in several ways. In particular, the college's support helped run the conference, and the NSF support provided student travel stipends and support for an invited speaker as well as a mentoring lunch.

This page intentionally blank.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Keynotes

A Quantum Probability Framework for Causal Inference

Jennifer S. Trueblood
Vanderbilt University

Reasoning about the causal relationships between events is an important component of cognition, allowing us to make sense of the world. Arguably, the most successful models of causal reasoning, causal Bayes nets, perform well in some situations, but there is considerable variation in how well they are able to account for data, both across scenarios and between individuals. More generally, decades of research have shown that human decision-making often violates the rules of classical (Bayesian) probability theory. Quantum probability (QP) theory provides an exciting new approach for modeling human cognition and decision-making.

In this talk, I will discuss how QP theory can be used to construct a framework for causal reasoning that accounts for behavior in situations where Bayes nets fail. I will discuss how changing assumptions about compatibility (i.e., how joint events are represented) leads to the construction of a hierarchy of models, from 'fully' quantum to 'fully' classical, that could be adopted by different individuals in different situations.

I will illustrate the approach with new laboratory experiments and model comparisons as well as discuss two factors that determine the form of the representation, individual differences in cognitive thinking style and familiarity with the causal reasoning domain. I will conclude by showing how the framework can used to understand real world causal judgments using a large (N=1200) experiment conducted during the US Presidential primaries involving judgments about the outcomes of primaries and the eventual nominations.

Automaton Theories of Human Sentence Comprehension

John T. Hale
Cornell University

The ability to understand what other people are saying, in a language that you know, is a impressive feat of cognition. Within this domain, many fundamental questions remain open. Among them: how does sentence structure figure in the comprehension process? Why is comprehension so fast & effortless most of the time? And which parts of the brain do which subtasks? This talk argues that cognitive architecture gives us a good head-start on these questions. By presenting a few proposals based on Hale (2014) it invites modelers to join in the enterprise.

Hale, J. (2014). *Automaton Theories of Human Sentence Comprehension*. Stanford, CA: CSLI Press.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Integrative Physiological Modeling: Looking at a Larger Picture

William A. "Drew" Pruett
University of Mississippi Medical Center

One approach to modeling is the use of minimal models that portray only the elements believed to be most causative of a particular phenomenon. An alternate approach is to connect many such minimal models together through their inputs and outputs to generate an integrated model in which larger phenomena can emerge. These emergent features do not belong to the minimal models, but rather are characteristic of their interactions. By integrating well-understood mechanisms into a consistent whole, the role of the individual pieces can be more fully understood. If the simple models and their linkages are viewed as the hypothesis of a theory, the integrated model is the testable part of that hypothesis.

Such models have been used to great effect in physiology to create cohesive scientific theories where no single causative agent could be found. Examples of this are the role of the kidney in establishing hypertension, and the complex interplay between the left and right heart in determining cardiac output. These models have been appreciated for this value for nearly 50 years in physiology, but enormous gaps remain to be studied. Among these is the relationship between cognitive state and physiological function.

In this talk, I will summarize past and current efforts in integrative physiological modeling from groups around the world, with special attention paid to the knowledge that flowed from studying the emergent properties of such models. Additionally, I will discuss domains in physiology that we believe will require cognitive models for deeper understanding of the physiology.

Tutorials

This page intentionally blank.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Stream: A Toolkit for Rapidly Developing High Precision Experiments

Gregory Wade (GWADE2778@gmail.com)
University of Delaware
Newark, DE 19716 USA

Brad Wyble (BWYBLE@gmail.com)
The Pennsylvania State University
Old Main, State College, PA 16801 USA

Keywords: Psychtoolbox; Stream Toolkit.

Introduction

With the increasing use of technology in the field of cognitive science there has been a need for software that can accomplish the tasks needed to further research. Many experiments require high levels of precision in spatial and temporal presentation of stimuli as well as the ability to collect data from participants. Many toolkits have been created for this purpose ranging different platforms and purposes. However, most psychologists lack the knowledge required to use these different toolkits, since they require substantial experience with programming languages like Python or MATLAB. Acquiring the necessary skills to use these toolkits is often practically impossible since the learning curve can be quite steep before one gets to the point of being able to use these tools effectively. The Stream Toolkit was created to bridge this gap. Stream simplifies the programming side of the experimental design process and allows people with a relatively basic level of programming knowledge to create complex experiments. Stream provides user-friendly scripts as well as many tutorials that will walk researchers through various aspects of experiment design, such as creating stimuli, displaying them, collecting responses, and analyzing data. It is still necessary to understand basic MATLAB functions and syntax in order to begin using Stream. The documentation provides a list of basic commands and topics that should be learned prior to beginning. Stream can be downloaded at: [https://bitbucket.org/streamtoolbox/stream_official_toolbox/downloads].

Psychtoolbox

Stream uses Psychtoolbox-3 (Brainard, 1997; Pelli, 1997; Kleiner et al, 2007) to interface with hardware, such as the graphics card, sound card, and mouse and keyboard drivers. This is essential to provide low-latency stimulus presentation and response collection. It is not necessary for the user to know Psychtoolbox, since Stream handles the interface. However, Psychtoolbox-3 must be installed and functional on the computer.

Summary of Features

Stream provides a skeletal framework of an experiment and a series of helper functions so that a user can add their own stimuli and data collection events. These helper functions include stimulus generation, event based presentation, collection of various data types, and simple analysis scripts. These features have been simplified for the user in order to streamline the experimental design process. The users only have to edit a few of the files in Stream while the brunt of the work happens behind the scenes in scripts that have already been written. All parts of Stream are open source, and users can access or modify Stream as needed.

Tutorials: A Good Place To Start

The Stream Toolkit provides users with a series of tutorials to teach the researchers how to use Stream. The main tutorials will walk users through the design of an experiment including data collection and analysis to become familiar with the format of Stream. Supplemental tutorials are provided for features not explained in the main tutorials. It is suggested that you read through the Main Stream Documentation before users begin the Stream Tutorials as the tutorials are meant to coincide with sections of the documentation.

Block Files

Block files are the main scripts that the user will edit in Stream. Block files represent the different experimental blocks in a task. When designing an experimental block, you will edit these files which will specify what stimuli are created, how they are to appear on the screen, and what response data are collected. If your experiment has multiple different blocks, you will create multiple block files that define each of them and run them in a specified order.

Stimulus Generation

At the top of your block file you will specify the stimuli that are used in the block. Stimuli are organized into 'sets'. You can create any number of stimulus sets that you choose, and each stimulus set can have as many stimuli as you choose. The only restriction on how you group stimuli into sets is that each set has to be comprised of the same kinds of stimuli (e.g. all of them are images). There are also a number of properties for each kind of stimulus that can be defined in order to customize your stimuli. By defining all of the stimulus properties in a structure you can bypass the need to understand Psychtoolbox functions, as Stream will do that work for you, although it will not hurt to become familiar with Psychtoolbox functions. Stream has many

different stimulus types including images, Psychtoolbox shapes, text, imagefonts, Gabor patches, and audio files.

Event-Based Presentation

All of the things that happen inside of a trial are called Events. Events are set up in block file and scheduled to happen at specific time points relative to the start of a trial. Once the events are scheduled, Stream takes over and executes them using a sophisticated timing loop that integrates stimulus presentation with data collection. All events are then time stamped with millisecond precision using the PsychToolbox GetSecs function. Screenshots can be collected at any point during the experiment, which is helpful for creating figures. Parallel port triggers can also be used in Stream.

Collecting response data

Responses are predefined using responsestructs, allowing you to give certain properties to a response event. These responsestructs are set up in the block files and then scheduled to occur at a particular time point. Any number of responsestructs can be used in an experiment. Stream allows for keyboard, mouse, and eye gaze responses (from Eyelink eye trackers).

Data Collection

When you run an experiment, Stream will create data files automatically. Stream is extensive in its data collection, such that every stimulus and event will automatically be saved along with timestamps. This extensive journaling allows for unanticipated exploratory analyses and also provides a safety net in case you forget to record condition labels. Because these files can be large and cumbersome to analyze, Stream also allows you to create compact data files containing only specific pieces of information that you choose. These compact files are a good way to filter only the information needed for analysis.

Analysis

You can use multiple methods to analyze your data, but if you choose, Stream has built in analysis scripts that will help you extract the data you have collected and allow you to perform analysis using custom code in MATLAB. Stream's analysis script is designed to pull information out of the compact data by looping through each subject, each block per subject, and each trial per block. Information from all of these trials is then copied into a structure. You can also opt to write these values to a text file that can be read into R, SPSS, or Excel if that is your preferred method of running statistical tests. Just like the block files, analysis scripts open-source and completely customizable.

Customer Support

For questions not covered in the documentation, customer support for the Stream Toolkit can be directed to Stream's main website [https://osf.io/tdvxm/]. Here you will find a Functions wiki as well as links to Google Group discussion forums where you can report bugs, suggest development projects, or as a general usage question.

References

Brainard, D. H. (1997) The Psychophysics Toolbox, *Spatial Vision 10*:433-436. http://psychtoolbox.org/credits/ [http://color.psych.upenn.edu/brainard/papers/Psychtoolbox.pdf

Pelli, D. G. (1997) The VideoToolbox software for visual psychophysics: Transforming numbers into movies, *Spatial Vision10*:437-442. http://www.psych.nyu.edu/pelli/pubs/pelli1997videotoolbox.pdf

Kleiner M, Brainard D, Pelli D, 2007, "What's new in Psychtoolbox-3?" Perception 36 ECVP Abstract Supplement.http://psychtoolbox.org/credits/[http://www.perceptionweb.com/abstract.cgi?id=v070821

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Integrative computational modeling of physiological and cognitive systems, and their interactions

Christopher L. Dancy (christopher.dancy@bucknell.edu)
Department of Computer Science, Bucknell University,
Lewisburg, PA 17837 USA

W Andrew Pruett (wpruett@umc.edu)
University of Mississippi Medical Center,
Jackson, MS 39216 USA

Keywords: Physio-cognitive models; cognitive architecture; ACT-R; HumMod; Hybrid Architecture

Introduction

The mind is embodied, physically and chemically, receiving and passing information to the body in myriad physiological feedback loops. The mind-body interface induces connections between different cognitive functions, and is an integral part of cognition. Understanding how physiological and cognitive mechanisms interact to influence behavior will require exploring the representative and systematic ways we can connect systems on the physiological and cognitive levels. As physiological sensors continue to become cheaper, more pervasive, and more accurate, computational cognitive modelers will have a unique opportunity to predict and explain human behavior using process models with representations on both the physiological and cognitive levels. This shift will result in models that more realistically operate over longer periods of time, allowing modelers access to more mechanistic models and predictions of behaviors given moderators like sleep deprivation, caffeine, or stress.

The ACT-R cognitive architecture (Anderson, 2007) can be used to model cognitive processes and their effects on behavior. However, ACT-R lacks a comprehensive way to simulate the effects of several cognitive moderators (Ritter et al., 2007) and the interactions of these moderators. The architecture has had its module functionality associated with certain areas of the brain and brain networks (e.g., see Anderson et al., 2008), this can make it more straightforward to understand where in the architecture certain moderators should affect behavior.

HumMod (Hester, Brown, et al., 2011) is a physiological model that simulates physiological systems from a middle-out perspective (i.e., see Hester, Iliescu, et al., 2011). The model integrates multiple tissue and organ level submodels along with a responsive cardiovascular system modulated by hormones and the autonomic nervous system. This allows one to explore the consequences of physiological perturbations (e.g., activation of nerves in the peripheral nervous system) both on the respective local systems and the overall global physiological system. However, as a stand-alone system, HumMod does not simulate high-level behavior.

ACT-R/Φ is a hybrid physio-cognitive architecture that combines the ACT-R and HumMod systems. The architecture can be used to explore interaction between physiological and cognitive systems and how these interactions modulate human behavior. A *physio* module controls the communication between the ACT-R and HumMod system, controlling the synchronization of the timing and also any bottom-up physiological modulation of cognitive processes. ACT-R/Φ has been used to explore several aspects of the physiology-cognition connection, including homeostatic drives (Dancy & Kaulakis, 2013), stress (Dancy, Ritter, Berry, et al., 2015) and sleep deprivation (Dancy, Ritter, & Gunzelmann, 2015).

In this tutorial, we will discuss physiological and cognitive processes, and interactions between systems at these levels, that are useful for modeling and simulating behavior on both the physiological and cognitive levels. We will use two representative systems (HumMod and ACT-R) as well as an integrated version of the two systems (ACT-R/Φ) to ground the discussed connections and interactions to a computational system. Tutees will then have the opportunity to build a hybrid computational physio-cognitive model, run the model in a simulated experiment, and interpret the predicted physiological and cognitive output against existing behavioral data.

About the Authors

Christopher L. Dancy is an assistant professor in computer science at Bucknell University and chair of the Behavioral Representation in Modeling and Simulation (BRiMS) society. His research interests focus on studying how physiology, affect, and cognition interact and what these interactions mean for memory, decision-making, and interfacing with systems. He uses computational process models and simulations, as well as experimental methods, to study these interactions and predict consequences for behavior.

W Andrew Pruett is an instructor in the Department of Physiology at the University of Mississippi Medical Center. His research concentration is *in silico* replication of clinical trials, especially with respect to hypertension. He uses population modeling and topological analytic techniques to advance the science of patient specific medicine.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York, NY: OUP.

Anderson, J. R., Fincham, J. M., Qin, Y., & Stocco, A. (2008). A central circuit of the mind. *Trends in Cognitive Sciences, 12*(4), 136-143.

Dancy, C. L., & Kaulakis, R. (2013). Towards adding bottom-up homeostatic affect to ACT-R. *In proceedings of the 12th International Conference on Cognitive Modeling*, Ottawa, Canada.

Dancy, C. L., Ritter, F. E., Berry, K. A., & Klein, L. C. (2015). Using a cognitive architecture with a physiological substrate to represent effects of a psychological stressor on cognition. *Computational and Mathematical Organization Theory, 21*(1), 90-114.

Dancy, C. L., Ritter, F. E., & Gunzelmann, G. (2015). Two ways to model the effects of sleep fatigue on cognition. *In proceedings of the 13th International Conference on Cognitive Modeling*, Groningen, Netherlands.

Hester, R. L., Brown, A. J., Husband, L., Iliescu, R., Pruett, D., Summers, R., & Coleman, T. G. (2011). HumMod: A modeling environment for the simulation of integrative human physiology. *Frontiers in physiology, 2*(12).

Hester, R. L., Iliescu, R., Summers, R., & Coleman, T. G. (2011). Systems biology and integrative physiological modelling. *Journal of Physiology, 589*(5), 1053-1060.

Ritter, F. E., Reifers, A. L., Klein, L. C., & Schoelles, M. J. (2007). Lessons from defining theories of stress for cognitive architectures. In W. D. Gray (Ed.), *Integrated Models of Cognitive Systems* (pp. 254-262). New York, NY: OUP.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

The Distributed Adaptive Control Architecture of the Embodied Situated Mind

Paul F.M.J. Verschure (paul.verschure@upf.edu)
Center of Autonomous Systems and Neurorobotics, University of Pompeu Fabra,
Catalan Institute of Advanced Studies (ICREA)
Barcelona, Spain

Keywords: Cognitive Architecture; Distributed Adaptive Control (DAC); Mind, Brain, Bode Nexus (MBBN).

Introduction

This tutorial introduces the Distributed Adaptive Control (DAC) theory of the principles underlying the Mind, Brain, Body Nexus (MBBN) that has been developed over the last 20 years (Verschure, 2003; Verschure, 2016). DAC assumes that the brain maintains stability between an embodied agent, its internal state and its environment through action. It postulates that in order to act, or know how, the brain has to optimize 5 fundamental objectives which can be labeled as: why, what, where, when and who. Thus the function of the brain is to continuously solve the so-called H5W problem with 'H' standing for the 'How' an agent acts in the world. The DAC theory is expressed as a neural-based architecture implemented in robots and organized in two complementary structures: layers and columns. The organizational layers are called: reactive, adaptive and contextual, and its columnar organization defines the processing of states of the world, the self and action or the interaction between the first two.

After an overview of the key elements of DAC, the mapping of its key assumptions towards the invertebrate and mammalian brain is described. The general overview of DAC's explanation of MBBN is combined with examples of application scenarios in which DAC has been validated, including mobile and humanoid robots, neuro-rehabilitation and the large-scale interactive space Ada. In this tutorial we will provide the elements necessary to implement an autonomous control system based on the DAC architecture and we will explore how the different layers of DAC contribute to solving a foraging task

Foraging is an advanced, goal-oriented behavior where prior knowledge of an environment and acquired behavioral strategies must be matched to the novelty and the hazards presented by an unpredictable world. DAC is based on the fundamental assumption that foraging can be explained on the basis of the interaction of three layers of control: reactive, adaptive and contextual. DAC was originally proposed as model for classical and operant conditioning. The reactive layer provides a set of reflexes allowing the system to interact with the environment – unconditioned stimuli to unconditioned responses. The adaptive layer is a model of classical conditioning and fulfills a twofold task. On the one hand it learns to associate the conditioned stimuli to the unconditioned responses, forming the conditioned responses. On the other hand, it forms internal representations of the conditioned stimuli, which are used by the contextual layer. We can define it as acquiring and shaping the agent-environment specific state space. The contextual layer is a model of operant conditioning providing the system with short and long term memory structures. The sensorimotor contingencies formed at the level of the adaptive layer are acquired and retained in these memory structures, forming behavioral sequences or policies. The representations stored in the contextual layer are constantly matched against the ongoing perceptions allowing for the retrieval of successful behavioral sequences in similar contexts.

The prototypical robot test case for DAC is a foraging task in an open arena. In this task, the robot, equipped with proximal and distal sensors, explores the arena in search of light sources while avoiding collisions with the surrounding wall. Colored patches on the floor serve as landmarks for navigation. In the framework of classical conditioning, the proximal sensors (e.g., distance and light) serve as aversive and appetitive unconditioned stimuli. Close to the light or when colliding with the wall an unconditioned response is triggered such that the robot approaches the light or turns away from the wall. The colored patches serve as conditioned stimuli. A visualization of such a task can be seen in Figure 1.

In this tutorial students will learn how to control the robot through the DAC architecture implemented using the IQR neuronal networks simulator (Bernardet et al., 2010) interfaced with the Gazebo robot simulator (Koenig et al., 2004) as seen in Figure 2. IQR implements the neuronal modules of the brain of the agent and Gazebo acts as the server of the simulated 3D environment. A VirtualBox Ubuntu virtual machine with a fully configured simulation setup and the DAC book (available at http://csnetwork.eu/CSN%20Book%20Series) accompany this tutorial.

Presenter

Paul Verschure is a professor at Universitat Pompeu Fabra, research professor at the Catalan Institute of Advanced Research and director of the Center of Autonomous Systems and Neurorobotics in Barcelona (Spain). His scientific aim is to find a unified theory of mind, brain and body through the use of synthetic methods and to apply such a theory to the development of novel cognitive technologies. Paul Verschure has pursued his research at different institutes in the US (Neurosciences Institute and The Salk Institute, both

in San Diego) and Europe (University of Amsterdam, University of Zurich and the Swiss Federal Institute of Technology-ETH and Universitat Pompeu Fabra in Barcelona). Prof. Verschure works on biologically constrained models of perception, learning, behavior and problem solving that are applied to wheeled and flying robots, interactive spaces and avatars. He maps these models to societal impact in the domains of health, cultural heritage and education. The results of these projects have been published in leading scientific journals including Nature, Science, PLoS, Neuron, Proceedings of the Royal Society and PNAS.

Figure 1: Gazebo robot simulator. Screenshot of Gazebo simulator showing the prototypical top view of a foraging task used to benchmark DAC with colored patches on the floor and a source of light represented by a bigger white patch on top of the screen.

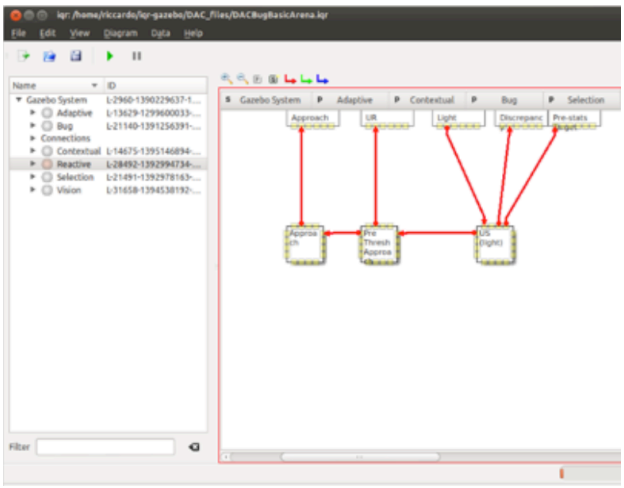


Figure 2: The IQR Neural Simulator. Screenshot of IQR showing a neural based implementation of the reactive layer.

References

Bernardet, U., & Verschure. P.F.M.J. (2010). "iqr: A tool for the construction of multi-level simulations of brain and behaviour." *Neuroinformatics 8.2*, 113-134.

Koenig, N., & Howard, A. (2004). "Design and use paradigms for gazebo, an open-source multi-robot simulator." Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on. Vol. 3. IEEE.

Verschure, P.F., Voegtlin, T., & Douglas, R.J. (2003). Environmentally mediated synergy between perception and behavior in mobile robots. *Nature*, *425*(6958), 620-624.

Verschure, P.F. (2016). Synthetic consciousness: the distributed adaptive control perspective. *Phil. Trans. R. Soc. B, 371*(1701), 20150448.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Tools for Cognitive Modeling: Developing tasks for universal access by models and human participants, exploring a massive parameter space to find the best fit of model to data, and analyzing the persuasiveness of the best-found fit

Vladislav "Dan" Veksler (vdv718@gmail.com)
U.S. Army Research Laboratory,
Human Research & Engineering Directorate
Aberdeen Proving Ground, MD, USA

Keywords: Simple Task-Actor Protocol (STAP); MindModeling.org; Model Flexibility Analysis.

Introduction

The aim of this tutorial is to walk participants through much of the cognitive modeling research cycle, from experiment/simulation development, to parameter exploration for finding the best fit of model predictions to empirical results, to determining the persuasiveness of the found fit (vis-a-vis Roberts & Pashler, 2000). This tutorial will provide hands-on experience with (1) Simple Task-Actor Protocol (STAP; Veksler, et al., in press) — a technology that enables reuse of task software for human participants in lab, online, and on mobile devices, and computational participants regardless of computational framework and programming language; (2) mindmodeling.org (Harris, 2008) — a free online parallel computing resource for exploring large parameter spaces; and (3) Model Flexibility Analysis (Veksler, Myers, & Gluck, 2015) — a method for estimating model complexity/flexibility.

Harris, J. (2008). MindModeling@Home: a large-scale computational cognitive modeling infrastructure. In The Sixth Annual Conference on Systems Engineering Research (pp. 246–252). Los Angeles, CA.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological review*, *107*(2), 358.

Veksler, V. D., Buchler, N., Lebiere, C., Morrison, D., & Kelley, T. D. (in press). The performance comparison problem: Universal task access for cross-framework evaluation, Turing tests, grand challenges, and cognitive decathlons. Biologically Inspired Cognitive Architectures.

Veksler, V. D., Myers, C. W., & Gluck, K. A. (2015). Model Flexibility Analysis. Psychological Review, 122(4), 755–769.

This page intentionally blank.

Long Papers

This page intentionally blank.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Cognitive Code: An Embedded Approach to Cognitive Modeling

Dario D. Salvucci (salvucci@drexel.edu)
Department of Computer Science, Drexel University
3141 Chestnut St., Philadelphia, PA 19104, USA

Abstract

For several decades, production systems have been the dominant framework in which (primarily) symbolic cognitive models have been developed. This paper proposes a different approach, *cognitive code*, in which behavioral models are developed directly in a modern programming language. However, unlike standard code, cognitive code has simulated timing and error characteristics intended to mimic those of human cognitive, perceptual, and motor processes. Some of the benefits of this new approach are illustrated in sample models of a paired-associates task, reading task, and dual-choice task.

Keywords: Cognitive architectures; ACT-R

Introduction

Since their introduction decades ago, cognitive architectures (Anderson, 1983; Newell, 1990; see also Gray, 2008) have provided a rigorous computational framework in which scientists can build and run cognitive models. Most importantly, a cognitive architecture represents a "unified theory of cognition" (Newell, 1990) that allows detailed exploration of the integration among various human systems, including cognitive, perceptual, and motor systems. Cognitive architectures have facilitated major advancements in cognitive science for specific research domains such as list memory (Anderson, Bothell, Lebiere, & Matessa, 1998) and multitasking (Borst, Taatgen, & van Rijn, 2010); at the same time, architectures have been applied to real-world domains such as gaming (Laird, 2002) and driving (Salvucci, 2006).

Even considering these successes, adoption of cognitive models outside of the academic research community[1] has been limited. There are arguably several reasons for this limited adoption:

Programming Paradigm and Language. Production systems have been the dominant framework in which (primarily symbolic) cognitive models have been implemented. Production systems, based on representation of processes as condition-action production rules, have been key to certain theoretical claims (e.g., learning from instructions: Taatgen, Huss, Dickison, & Anderson, 2008). However, from the perspective of programmers outside the cognitive-architecture community, production systems are largely an unknown quantity; instead, modern programmers typically develop code using modern procedural and object-oriented programming languages (Java, C++, Python, etc.). Learning the very different programming paradigms and patterns used in production systems is a significant barrier to developing models, even for those with a significant computer-science or programming background.

Model-Centered Development. For those in the research community, the cognitive model is often the centerpiece of the main research effort, and a great deal of time and care is taken to develop these models. For this reason, modeling frameworks often include an integrated user-interface environment and suite of tools to facilitate model development. However, for an outsider looking to embed a cognitive model into their own project—say, a game-engine programmer who wants to develop a cognitive agent to embed into a larger game—the model is a peripheral component rather than a central one. For this audience, learning an entirely new modeling language and development environment can be a large investment, often too large to make it worthwhile.

Lack of Model Integration. Cognitive modelers, especially those interested in cognitive architectures, have long stressed the benefits of a community-driven approach to unified theories of cognition. Over the years, this integration has largely come about at the architectural level, with a wide set of models using a single architecture or framework to generate behavior. Unfortunately, integration among models themselves—for example, reuse of existing models to develop new models—has arisen much less frequently, partly because modeling frameworks have not emphasized rigorous formal APIs that are crucial for integration.

Modern cognitive architectures and models have much to offer beyond the boundaries of the research community; with a blend of psychological theory and computational simulation, they contain a breadth and depth of predictive accounts that can be widely useful for other research and practical domains. Yet, because of the reasons above, the investment needed to extracting predictions from these models is often too large for those with less than a primary interest in cognitive modeling.

[1] Companies such as Carnegie Learning and Soar Technology have successfully applied cognitive models beyond the walls of the research community; still, the impact of cognitive modeling pales in comparison to related research methodologies—for instance, machine learning—that have exploded in popularity.

Cognitive Code

Cognitive code is code embedded in a modern programming language that aims to simulate and mimic human cognitive, perceptual, and motor processes. Of course, cognitive models and architectures have long used computational representations to simulate human processes, but they have (as noted earlier) generally defined their own programming language for this purpose, and generally relied on production systems as a central tenet of their representations. In contrast, cognitive code is embedded and written directly in an existing programming language, using structures and patterns already familiar to most programmers. At the same time, cognitive code differs from standard code in that it includes new software design patterns and libraries to facilitate the simulation of timing and errors inherent in human processes.

Before delving into the details, let us illustrate the basic concept with a simple example. Consider sample cognitive code (here in Java) that stores a new piece of information into memory:

```
memory.store(new Chunk("cat")
     .set("owner", "Jane")
     .set("name", "Whiskers"));
```

Here we create a "chunk" of information (as in ACT-R: Anderson, 2007) that defines knowledge about Jane's cat, and we store it into memory. Later, we try to recall this information from memory with a query:

```
Chunk chunk = memory.recall(new Query("cat")
     .add("owner", "Jane"));
```

In each case, the code uses patterns that would be straightforward and recognizable by most programmers. At the same time, this code conceptually differs from standard code in two ways. First, each step incurs a simulated temporal cost that corresponds to the time needed to perform this action in the cognitive system. In our example, the `memory.store()` action incurs some time, say a few hundred milliseconds, to add this knowledge to memory, and the `memory.recall()` action also incurs an amount of time that may depend on many factors (time since learning, number of times practiced, etc.). Second, each step has some potential for failure that mimics a true cognitive system; for example, the `memory.recall()` action could fail and return a *null* result depending on the current state of the cognitive system.

We now describe a prototype system that embodies the cognitive code approach. The system is implemented in Java as a library that can be readily integrated into other projects.

Core Simulation System

The core system centers on the concept of an *agent* that acts within the simulated world. An agent consists of any number of *modules* that define behavior for a particular subsystem (e.g., memory, vision, etc.) or for a particular domain (arithmetic, driving, etc.). The basic unit of information shared across modules is an *item*, which comprises a set of slot-value pairs, one of which can be the "isa" type of the item. Modules can utilize *workers* to perform work for them, and can share information with other threads through *buffers*.

Because the passage of time is central to the cognitive code approach, we also require some way to simulate a central clock within the code. Within a single thread, we simply maintain a simulated time to be incremented by each cognitive step. In the general case, though, we need multiple threads to share a single clock; human multitasking will be best represented as separate cognitive threads, as discussed later, and even a single thread may require that certain operations happen in parallel (e.g., moving a hand while recalling information).

The core simulation enables the central clock as follows. The system assumes that individual operators on a particular thread can define their own delays, effectively stopping execution until the clock reaches the new time after delay. For example, let's say we run the following code on one agent thread:

```
agent.wait(1.0);
memory.store(new Chunk("cat"));
```

and run the following code on another agent thread:

```
agent.wait(2.0);
Chunk chunk = memory.recall(new Query("cat"));
```

The first thread requests that time advance to 1.0 seconds, while the second thread requests an advance to 2.0 seconds. In this situation, the first thread succeeds in advancing the clock to 1.0 seconds, and then performs the memory store. The second thread's *wait* step will block until 2.0 simulated seconds have passed, guaranteeing that the subsequent *recall* step will happen only after the first thread's *store* step.

The underlying implementation of the shared clock uses a type of cyclic barrier (Java's Phaser class) to ensure that all concurrent threads are synchronized as each reaches a new time step. If desired, the system can be run in real time such that the clock corresponds to the actual time (or a multiple thereof). In typical usage, however, simulated time does not need to correspond to real time; in fact, it is often advantageous to run the simulation as quickly as possible, and then examining time after the fact for various purposes (e.g., to predict how long a particular set of actions might take).

Memory System

The memory system is based wholly on the ACT-R theory of declarative memory (Anderson, 2007). In this theory, memory consists of *chunks* of knowledge with slot-value pairs, which over time strengthen or decay with practice or lack of use. In particular, each chunk has an associated *activation* that defines how readily the chunk can be recalled, as dependent on its prior usage: if a chunk is "used" (recalled or re-stored) at times t_k, the activation A_i for chunk i can be defined as

$$A_i = ln\left(\sum_k t_k^{-d}\right)$$

where d is the memory decay factor that determines how quickly the usages decay. Given the chunk's activation, we can determine the probability to recall the chunk as

$$Pr\ (recall) = \frac{1}{e^{-(A_i-\tau)/s}}$$

for a retrieval threshold τ and noise level s. If the chunk can be successfully retrieved, the time needed for retrieval is defined as

$$T_{recall} = Fe^{-A_i}$$

In our cognitive code system, chunks are stored using the `memory.store()` action, which incurs only a 50 ms time for a cognitive step—the same 50 ms delay used in many cognitive architectures for the firing time for a production rule (e.g., ACT-R, Soar). For recall, our system includes two types of actions. First, there is a non-blocking action `memory.startRecall()` that initiates the recall action but allows the code to continue past this point after a 50 ms cognitive step time. A non-blocking action of this type allows the code to continue and perform other actions while recall is taking place (e.g., watching for a visual stimulus or typing a key). The `memory.getRecalled()` action is then used to access the recalled information, and this command blocks until the recall is complete. Second, there is the blocking command we saw earlier, `memory.recall()`, which is equivalent to performing a `memory.startRecall()` followed by a `memory.getRecalled()`. Both types of recall actions take as an argument a query that partially defines the desired chunk (as seen earlier with the "cat" example), and both can fail and return *null* if the chunk is not successfully recalled.

Chunk rehearsal—that is, the usages defined earlier—can take several forms. When a chunk is initially created, this is defined as its first use. If an exact copy of this chunk is stored later, the copy is merged into the original chunk and becomes another use of the chunk. Finally, any recall of the chunk serves as a rehearsal and adds to the use count. Thus, all of these forms contribute to the gradual increase in a chunk's activation; in contrast, the lack of use causes any early uses to decay away, making the chunk more difficult to recall and more costly (in terms of time) if successfully recalled.

Perceptual System

The perceptual system is primarily based on ACT-R and partly based on a related theory of eye movements. The vision module, following ACT-R, assumes a spotlight of visual attention that moves according to a two-stage *where-what* process of finding objects and encoding objects. The non-blocking action `vision.startFind()` attempts to find a visual location that matches the given query based on perceptual features available in peripheral vision (like position, color, etc.), and its complementary command `vision.getFound()` returns the found location. Their blocking counterpart `vision.find()` achieves the same effect in a shorthand simpler action. Another command, `vision.waitFor()`, waits until a location matching the query appears in view (e.g., to model waiting for a visual stimulus). Once a location is found, there are analogous actions for encoding the object at the location and returning information about the object: the non-blocking action `vision.startEncode()` and its associated action `vision.startEncode()`, along with their blocking counterpart `vision.encode()`.

The movement of visual attention from one object to another generates movements of the system's simulated eyes as defined by the EMMA theory (Salvucci, 2001), which in turns derives from the E-Z Reader theory of eye movements in reading (Reichle, Pollatsek, Fisher, & Rayner, 1998). This provides the system with a powerful predictive dimension: the code never explicitly moves the eyes, but in moving attention, the eyes follow and demonstrate several interesting aspects of eye-movement behavior: the time lag between a movement of attention and a movement of the eyes; the possibility of skipping over an encoded object when that object is easy to encode (e.g., a high-frequency word); and the possibility of re-fixating an encoded object with multiple eye fixations when that object is difficult to encode (e.g., a low-frequency word).

In addition to vision, the system includes audition to model detection and encoding of aural information. Specifically, the audition module includes `audition.startDetect()` and `audition.detect()` actions (non-blocking and blocking, respectively) to detect a sound, and an associated action `audition.waitFor()`that waits for the next sound that matches a query. It also includes `audition.startEncode()` and `audition.encode()` actions that are analogous to these actions in the vision module.

Motor System

The motor system is currently focused on using a mouse and keyboard in a desktop computer environment. The mouse-movement module uses Fitts' law in the same manner as ACT-R and EPIC (Meyer & Kieras, 1997). The module includes the expected actions to move and click the mouse: `mouse.startMoveTo()` and `mouse.startClick()` as non-blocking actions, and `mouse.moveTo()` and `mouse.click()` as blocking actions.

For typing, the motor system is based on the TYPIST model (John, 1996). Typing is invoked with a `typing.type()` action that specifies the text to output. The typing module breaks up the given text into words and then types each word as a 50 ms cognitive step followed by the execution of the motor actions for each keystroke; shifted keys (e.g., capital letters) require a keystroke for the *shift* key before the keystroke for the character. The time for each keystroke was estimated as a function of typing speed as measured in gross words per minute. Specifically, a function was fit to Figure 4 of John (1996) to yield the following estimate of keystroke time T_{key} as a function of words per minute wpm:

$$T_{key} = .0000083(wpm)^2 - .003051(wpm) + .31727$$

The typing module can thus be set to any typing speed between 30 and 120 words per minute for an estimate of typing times at that speed.

```
[1]   String word = (String) vision.encode(vision.waitFor(new Query("word")));
[2]   Chunk chunk = memory.recall(new Query("pair").add("word", word));
[3]   if (chunk != null)
[4]       typing.type(chunk.getString("digit"));
[5]   int digit = (Integer) vision.encode(vision.waitFor(new Query("digit")));
[6]   memory.store(new Chunk("pair").set("word", word).set("digit", digit));
```

Figure 1: Cognitive code for the paired-associates task.

The other component of the motor system is the speech module, as represented by a `speech.say()` action that takes a simple string as input. Roughly like the ACT-R speech system, this module breaks the string into syllables and outputs the speech with a delay equal to a base time (200 ms) plus an execution time per syllable (150 ms). Whereas ACT-R has a simple assumption of syllables (one syllable per 3 characters), the module here uses a more complex method to break up syllables according to a number of rules for English pronunciation tested on a small corpus of common words.

Cognitive Code as Software

The fact that cognitive code is implemented as software in a mainstream programming language lends it several benefits over cognitive models developed in a typical cognitive architecture approach. In contrast to a monolithic cognitive architecture, modules in cognitive code are instantiated as needed, and using different implementations of a particular type of module does not pose a problem. For example, the following code defines a new agent, the eyes of the agent, and finally the vision module:

```
Agent agent = new Agent();
Eyes eyes = new Eyes(agent);
Vision vision = new Vision(agent, eyes);
```

Various methods for these components are easily accessible: for instance, a programmer might check aspects of the agent (e.g., `agent.getTime()`), move the eyes to a new location (`eyes.moveTo()`), or change parameters of the visual system (`vision.setFindTime()`). Developers can extend objects to include additional functions, or can build new objects that use cognitive-code objects as primitives. In fact, as the approach evolves, we expect different implementations of the modules to provide alternative theoretical approaches—in this case, say, we might have a different visual system based on pixels and salient features.

Another large benefit is that cognitive code inherits the many structures and tools already used by modern software for developing, interfacing, and testing code. Instead of having specialized IDEs (integrated development environments), cognitive code allows a programmer to use their preferred IDE for development. Cognitive code also inherits the robust APIs (application program interfaces) of modern programming languages, such as packages, classes, interfaces, and related constructs—a big advantage in accessing others' code and successfully integrating it with one's own code. Finally, cognitive code can be tested rigorously utilizing the same tools commonly in use today (e.g., JUnit tests in Java). In this case, each test can check not only whether the code runs correctly as a piece of software, but also whether some cognitive code fits an appropriate empirical (human) data set; in other words, the code's correctness also depends on whether it accurately mimics the behavior of human behavior in the chosen domain.

Illustrative Examples

We now provide a few illustrative examples of cognitive code, all of which represent re-implementations of existing models developed in a cognitive architecture. Our goal here is to demonstrate that cognitive code can produce much the same behavior and predictions as architecture models, but the code blocks that generate these behaviors are simpler and more learnable than their architectural counterparts.

Paired Associates

One of the standard models in Unit 4 of the ACT-R tutorial[2] is a model of the paired-associates task. In this task, participants see a word stimulus (e.g., "king") and must type a digit that is associated to that word in the experiment (e.g., "7"). The participant does not know the associations at the outset, but over time, they learn them and gradually become better at recalling the associated digit, improving their correctness and (for correct responses) improving their response times. Students studying the ACT-R architecture might model this task as they learn to understand ACT-R memory theory and implement their first models of memory storage and recall. Even after a few prior lessons in the syntax and semantics of the ACT-R modeling language, the paired-associates model can be difficult for students to understand and might take a typical student one to a few hours to work through and understand.

A cognitive-code model of the paired-associates task is shown in Figure 1. Starting at line 1, the model first waits for a word, blocking on the `vision.waitFor()` command until the stimulus appears, and then encodes the word. It then tries to recall a chunk that represents the word-digit pair in memory; if the chunk is successfully recalled, the model types the digit as a response. The model then waits for the digit (which appears in all cases) and stores the word-digit pair to memory. (Note that if the word-digit pair is already in memory, this strengthens the pair chunk as described earlier.) This model behaves essentially the same as the ACT-R tutorial model, and successfully produces the behavioral

[2] http://act-r.psy.cmu.edu

```
[1]   Visual visual = vision.find(new Query("word"));
[2]   while (visual != null) {
[3]       vision.encode(visual);
[4]       agent.wait(MEMORY_RECALL_DURATION);
[5]       visual = vision.find(new Query("word").add(Visual.SEEN, false));
[6]   }
```

Figure 2: Cognitive code for the reading task.

```
[1]   agent.run(() -> {
[2]       Object tone = audition.encode(audition.waitFor(new Query("tone")));
[3]       if (tone.equals("low"))
[4]           speech.say("low");
[5]   });
[6]   agent.run(() -> {
[7]       Object stimulus = vision.encode(vision.waitFor(new Query("stimulus")));
[8]       if (stimulus.equals("O--"))
[9]           typing.type("1");
[10]  });
[11]  agent.wait(1.0);
[12]  vision.add(new Visual("stimulus", 10, 10, 10, 10), "O--");
[13]  audition.add(new Aural("tone"), "low");
[14]  agent.waitForAll();
```

Figure 3: Cognitive code for the dual-choice task: blue for the aural-vocal task, green for the visual-manual task.

```
0.000   agent              wait for {isa=stimulus}
0.050   agent              wait for {isa=tone}
1.000   agent.vision       found {isa:"stimulus" x:10 y:10 w:10 h:10 seen:false}
1.000   agent              encode {isa:"stimulus" x:10 y:10 w:10 h:10 seen:false}
1.050   agent.audition     found {isa:"tone" heard:false}
1.050   agent              encode {isa:"tone" heard:false}
1.185   agent.vision       encoded O--
1.185   agent              type "1"
1.235   agent.hands        typing "1"
1.385   agent.audition     encoded low
1.385   agent              say "low"
1.435   agent.speech       saying "low"
1.444   agent.hands        typed 1
1.785   agent.speech       said "low"
```

Figure 4: Trace of the cognitive code in Figure 3: blue for the aural-vocal task, green for the visual-manual task.

patterns exhibited by people, namely the increased accuracy and decreased response time with practice.

Upon learning the cognitive-code approach, arguably the most difficult aspect of this code is learning the way that timing works—especially understanding that some actions will block until a stimulus appears or until a chunk is recalled. In general, though, a programmer versed in Java can easily understand the control flow here, and knows how to get this code to compile correctly and how to access API documentation when needed. (For example, we have not explained the details of the *Query* class used in the visual and memory requests, but these details are easily discovered in the documentation through a modern IDE.)

Reading

As part of a validation of the EMMA model of eye movements, Salvucci (2001) described a parsimonious model of sentence reading that simply encoded words from left to right (ignoring any deeper understanding to focus on the eye movements themselves). This model was a test of EMMA's ability to predict eye movements directly from straightforward shifts of visual attention, examining measures of gaze durations, first-fixation durations, and skip probabilities as a function of word frequency.

A similarly straightforward snippet of cognitive code that performs sentence reading is shown in Figure 2. The code implements a loop that iteratively finds and encodes each word. The find actions in lines 1 and 5 utilize the property of the visual system that, by default, vision finds locations closest to the current eye location; in line 5, the find command also makes sure to find a word that has yet to be seen. When a "visual" is found, the model encodes the contents of the word and fakes the semantic processing of the word by simply waiting for some time delay intended to mimic a lexical retrieval. Of course, a more rigorous model of reading would need to flesh out this aspect of the code, but for now, this code is sufficient to move attention along from one word to the next, triggering the predictions of the EMMA model and its resulting eye movements. The behavior of this model fits well to the empirical data, with correlations above .95 and low errors for the three measures mentioned above.

When developing such a model in a production-system architecture, the control flow of the model can be very difficult for students to grasp—both the flow within an individual production rule and the higher-level control flow among the production rules. In contrast, the iterative loop here is a familiar construct to programmers, and more clearly demonstrates the simplicity of the reading model and thus the predictive power of the underlying model of eye movements.

Dual Choice

Human multitasking has been characterized as the interaction of separate *cognitive threads* that interleave their processing (Salvucci & Taatgen, 2008, 2011). Figure 3 shows a simple example of how threading would work in a cognitive-code approach, illustrating a model of a dual-choice task that has been characterized as "perfect time-sharing" (Schumacher et al., 2001). This code starts two threads (via the `agent.run()` command): the first thread (lines 2-4, blue text) listens for a tone and then generates a speech response; and the second thread (lines 7-9, green text) waits for a visual stimulus and then generates a keystroke response. The task code (lines 11-14) waits 1 second, presents simultaneous visual and aural stimuli, and finally waits for all threads to complete.

The resulting simulation trace of this model, including timing (left-most column), is shown in Figure 4, with trace events color-coded as belonging to the first thread (blue) or the second (green). Both threads start waiting for their respective stimuli at the outset of the simulation. When the stimuli appear at the 1.0-second mark, the second thread sees the visual stimulus and begins to encode it; meanwhile, the first thread requires 50 ms to detect the aural stimulus, and after this delay it also encodes the sound. The motor responses—typing for the second thread, speech for the first—overlap such that neither thread experiences any time delays. Thus, the overall trace closely resembles the kind of perfect time-sharing behavior exhibited by more complex ACT-R models of this task (e.g., Salvucci & Taatgen, 2008).

Discussion

Cognitive code aims to strike a balance between the theoretical rigor of modern cognitive architectures and the practicality of modern programming languages and environments. The above examples show how concepts of cognitive code can lead to much simpler models, especially when compared with production-system architectures, and especially for the typical programmer versed in procedural and object-oriented languages commonly in use today.

There are at least two limitations of cognitive code compared to production systems that should be noted. First, production systems have the potential to be more flexible in their flow of control, whereas the procedural code here has a more rigid sequencing of actions. However, one might argue that most production-system models do not exploit this flexibility, but instead constrain the rules to embody the same kind of procedural control flow as the cognitive code here. Second, production systems allow for learning of the rules themselves (e.g., ACT-R's production compilation), whereas cognitive code is fixed by the developer. Allowing for procedural learning through cognitive code is still under exploration, but for now, this is perhaps its biggest limitation compared to production systems. Nevertheless, we remain hopeful that the benefits of the cognitive-code approach will ultimately pay dividends in expanding the usability and learnability of cognitive modeling to a wider audience.

References

Anderson, J. R. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* New York: Oxford University Press.

Anderson, J. R., Bothell, D., Lebiere, C., & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language*, *38*, 341-380.

Borst, J.P., Taatgen, N.A., & van Rijn, H. (2010). The problem state: A cognitive bottleneck in multitasking. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, *36*, 363-382.

Gray, W. D. (2008). Cognitive architectures: Choreographing the dance of mental operations with the task environment. *Human Factors*, *50*, 497-505.

John, B. E. (1996). TYPIST: A theory of performance in skilled typing. *Human-computer interaction*, *11*, 321- 355.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review*, *104*, 3-65.

Laird, J. E. (2002). Research in human-level AI using computer games. *Communications of the ACM*, *45*, 32-35.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, *33*, 1-64.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Reichle, E. D., Pollatsek, A., Fisher, D. L., & Rayner, K. (1998). Toward a model of eye movement control in reading. *Psychological Review*, *105*, 125-157.

Salvucci, D. D. (2001). An integrated model of eye movements and visual encoding. *Cognitive Systems Research*, *1*, 201-220.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors*, *48*, 362-380.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review*, *115*, 101-130.

Salvucci, D. D., & Taatgen, N. A. (2011). *The Multitasking Mind*. New York: Oxford University Press.

Schumacher, E. H., et al. (2001). Virtually perfect time sharing in dual-task perfor- mance: Uncorking the central cognitive bottleneck. *Psychological Science, 12*, 101–108.

Taatgen, N. A., Huss, D., Dickison, D. & Anderson, J. R. (2008). The acquisition of robust and flexible cognitive skills. *Journal of Experimental Psychology: General*, *137*, 548-565.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Microgenetic Analysis of Learning a Task: Its Implications to Cognitive Modeling

Jong W. Kim (jong.kim@ucf.edu)
University of Central Florida
Orlando, FL 32816 USA

Frank E. Ritter (frank.ritter@psu.edu)
College of Information Sciences and Technology, Pennsylvania State University
University Park, PA 16802 USA

Abstract

We report a microgenetic and quantitative analysis of a large learning data set. We analyzed performance change by four practice trials (once per day) and by the 14 different subtasks with more than 500 total keystrokes. Specifically, we compared performance change across the subtasks—some subtasks are cognitive problem-solving and others are perceptual-motor driven tasks. This microgenetic approach provides an understanding of how a local performance in a task affects the global performance of a whole task. We computed the learning curve constants for the different subtasks. We found evidence to support the KRK theory of learning and retention (Kim & Ritter, 2015). The results suggest that learning varies by subtask and by its type.

Keywords: Microgenetic analysis; Learning; Cognitive modeling.

Introduction

In general, learning can be described as a speed-up or practice effect (Ritter, Baxter, Kim, & Srinivasmurthy, 2013; Seibel, 1963). To help better understand our learning performance, it is necessary to focus on a couple of variable factors in tasks and their types. Complex tasks may consist of different components of subtask skills. Presumably, different subtask skills may be learned and retained in our memory. This understanding would affect the perspectives of learning, learning environments, instructional systems (e.g., contents), and interface design of such systems.

As one small step contributing to learning research, we investigated learning and retention of a complex task consisting of the 14 subtasks by comparing two input modalities (Kim & Ritter, 2015). This investigation suggests that the prevalence of GUI interfaces can be attributed to a more relearnable design compared with a keystroke-based interface, and suggests more investigation on where learning (and forgetting) occur during the course of complex tasks.

In this paper, we conduct a deeper analysis; a microgenetic analysis of learning in an attempt to identify how learning is different across the 14 subtasks. We look at individual subtask skill components over four practice trials. This approach is similar to a microgenetic study examining sources of change in cognitive development and learning (e.g., Agre & Shrager, 1990; Moon & Fu, 2008; Siegler, 2006). It is expected that our approach can provide a deeper understanding of where learning occurs and how different knowledge types are learned.

Learning as a Whole Task

A considerable amount of literature suggests a consensus understanding of learning; a three-stage process of learning provides a theoretic account of performance change including (a) acquiring declarative knowledge from instruction to perform a task in the first stage, (b) consolidating the acquired knowledge into a procedural form with practice in the second stage, and (c) tuning the knowledge toward overlearning exhibiting the speedup effect of the knowledge application mental process (Anderson, 1982; Fitts, 1964; Rasmussen, 1986). Based on this consensus foundation of learning, a study of forgetting expands how an individual learns and retains knowledge and skills theoretically, empirically, and computationally (Kim & Ritter, 2015), shown in Figure 1. A widely used cognitive architecture, ACT-R, implements the computational features of the three-stage process by proposing that performance change follows a regularity known as the power law of practice—the time to complete a task speeds up with practice according to a power function (e.g., Anderson, Fincham, & Douglass, 1999; Newell & Rosenbloom, 1981; Seibel, 1963). An exponential function is also widely accepted to summarize the practice effect (e.g., Heathcote, Brown, & Mewhort, 2000).

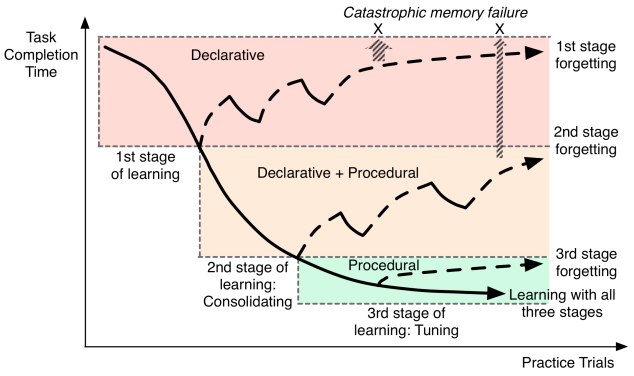


Figure 1: The KRK three-stage learning and retention theory (Kim & Ritter, 2015).

Recent reports provide a predictive analysis of a spreadsheet task, the Dismal spreadsheet task (Kim & Ritter, 2015; Paik, Kim, Ritter, & Reitter, 2015), using KLM-GOMS (Card, Moran, & Newell, 1983) and ACT-R (Anderson, 2007). These analyses examine performance change from a novice through an intermediate to an expert

performing a complex task. The task includes subtasks, and the time to complete the task is predicted by the aggregate resources subtasks use (i.e., cognitive, perceptual, and perceptual-motor skills). These predictions can be meaningfully decomposed to each subtask skill and single action, and these can be compared with the data on the same level, providing an organization for a microgenetic analysis.

A Baseline Prediction

As a baseline prediction, a KLM-GOMS model was used to predict error-free expert performance on the Dismal task. The task completion times were computed to be compared with ACT-R predictions and the human data of the whole task of the Dismal spreadsheet task. The model includes primitive physical-motor operators (K – keystroke, P – pointing, H – homing, and D – drawing), mental operators (M), and system response time (R), as shown in Equation (1). In the interest of simplicity and because of relatively fast response times, we ignored the system response time (T_R=0).

$$T_{execute} = T_K + T_P + T_H + T_D + T_M + T_R \quad (1)$$

We used three physical-motor operators (K, P, and H) and the mental operator (M) for time predictions of the Dismal spreadsheet task. The default time was used for homing and mental operators. During the mental operator time (T_M), participants mentally prepare what to press and retrieve items from memory including the next step. We followed the existing heuristic rules for determining the use of mental preparation (Card, Moran, & Newell, 1983, p. 265) and used the default time, 1.35 s. We placed a mental operator in front of all pointing activities (pointing to a menu item) and all key-press activities (pressing a keystroke command). To complete the first subtask (Open File), theoretically, participants in the keyboard group needed 3 mental operators (refer to Table 2). The homing time (T_H) for hand movements between different physical devices was 0.4 s.

To calculate the keystroke time (T_K), we know that it varies across individuals. We, therefore, computed the time from the first keystroke to the last in the first subtask for both modalities. The average keystroke time ranged from 0.95 s/keystroke on the first day of learning to 0.47 s/keystroke on the last day of learning. If we refer to the keystroke time in Card et al. (1983, p. 264), our data indicate the participants' keystroke speed resided between the worst typist, 1.20 s and the speed of average non-secretary typist, 0.28 s. We used 0.47 s for the T_K parameter as an expert performance. Shift and control keys were counted as a separate keystroke. The predicted task completion time for users in the keyboard group was 666.67 s as seen in Table 2. We present the details of the KLM-GOMS analysis of each subtask in the Microgenetic Analysis of Learning section.

ACT-R Prediction

Several cognitive architectures predict learning, which is beyond the capability of KLM. Particularly, the ACT-R architecture provides predictions of performance changes due to learning. Furthermore, the ACT-R model can predict learning on this task from a novice to an expert, as shown in Figure 2. The model consists of production rules and declarative memory elements to represent practice effects, which can be compared with human learning data (Paik et al., 2015).

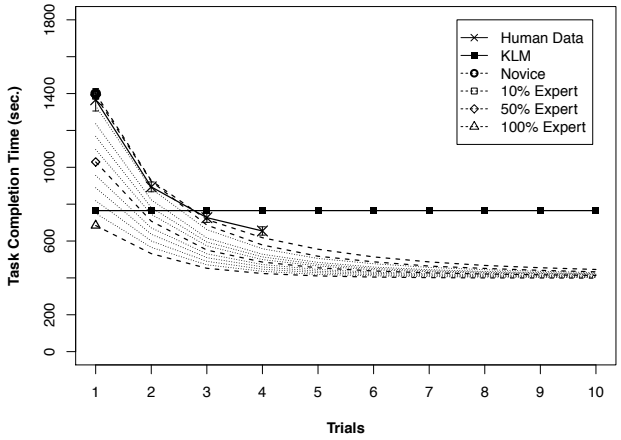


Figure 2: ACT-R models of the Dismal task (dashed lines, from fully novice to previously practiced expert), along with human aggregate data (X's and SEM error bars), and the KLM prediction (solid line) (taken from Paik et al., 2015).

The Task and Data

The task that we apply a microgenetic approach to is a large complex office-related task, the Dismal spreadsheet task (Kim & Ritter, 2015). Dismal is a spreadsheet that runs within Emacs and was initially developed to analyze behavioral process models and data (Ritter & Larkin, 1994; Ritter & Wood, 2005).

The Task

The subtasks include: opening a spreadsheet file, saving the file as another name, and completing a complex spreadsheet manipulation by calculating and filling in several blank cells, including five data normalization calculations, five data frequency calculations, ten calculations of length, ten calculations of total typed characters, four summations of each column, and an insertion of two rows to type in the current date and name using Dismal keystroke commands. Together, they can be grouped into the 14 subtasks, as shown in Table 1. More information about the task (e.g., the task environment and the procedure) is available (Kim & Ritter, 2015).

Table 1: The subtasks in the Dismal spreadsheet task.

Subtasks	Keystrokes
(1) Open File	Press C-x C-f Type <normalization.dis> ↲
(2) Save As	Press C-x C-w Type JWK.dis ↲
(3) Calculate Frequency (B6 to B10)	Move the point to B6 by using C-p, C-n, C-f, or C-b Press e Type "(/ (* c6 b12) 100.0)" ↲ Repeat for B7 to B10
(4) Calculate Total Frequency (B13)	Move to the point to B13 Press **e** Type "(dis-sum b1:b10)" ↲
(5) Calculate Normalization (C1 to C5)	Move the point to C1 Press e Type "(/ (* 100.0 b1) b12)" ↲ Repeat for C2 to C5
(6) Calculate Total Normalization (C13)	Move the point to B13 Press **e** Type "(dis-sum c1:c10)" ↲
(7) Calculate Length (D1 to D10)	Move to the point D1 Press e Type "(length a1)" ↲ Repeat for D2 to D10
(8) Calculate Total Length (D13)	Move the point to D13 Press e Type "(dis-sum d1:d10)" ↲
(9) Calculate Typed Characters (E1 to E10)	Move the point to E1 Press e Type "(* b1 d1)" ↲ Repeat for E2 to E10
(10) Calculate Total Typed Characters (E13)	Move the point to E13 Press e Type (dis-sum e1:e10) ↲
(11) Insert Two Rows	Move the point to A0 Press C-u type 2 i r ↲
(12) Type in *Name* (A0)	Press e Type in *Name* ↲
(13) Insert Current Date (A1)	Move the point to A1 Press e Type "(dis-current-date)" ↲
(14) Save As Printable Format	Press C-x C-w Type <normalization-initials.dp >↲

The Data

The data used in this paper is 30 participants' learning performance. A learning session consists of a study session and a test trial. A study session is when a participant used the study booklet to learn. Each study session is limited to 30 minutes of study. A test trial in the learning session is when participants perform the given tasks without the study booklet.

In the first week, participants performed four consecutive learning sessions. On Day 1, participants had a maximum of 30 minutes to study the spreadsheet tasks and then performed the tasks. On Days 2 to 4, participants were allowed to refresh their acquired knowledge from Day 1, using the study booklet, and then performed the tasks.

The task completion time and every keystroke movement were measured by the Recording User Input (RUI) system (Kukreja, Stevenson, & Ritter, 2006). The target participants in this report used a keystroke-based interface to complete the task. The raw data included every keystroke and its time (in ms). This allows us to investigate performance change on a more microgenetic level by examining the time to perform each subtask and unit task during the practice trials.

Microgenetic Analysis of Learning

We next describe the subtasks and then how learning happens by subtask.

Preliminary Analysis of the Subtasks

Table 2 shows the KLM actions in the task based on the instructions. We initially analyzed the recorded performance under the KLM framework as seen in Table 2.

Each subtask has different mental and keystroke operators. The KLM analysis is based on the number of each operator in each subtask according to Eq. 1. It provides us with a basic quantitative baseline prediction of user performance, not performance change. Three practice trials is enough to get to the KLM times. With even more practice performance is faster than the KLM predictions (Card, Moran, & Newell, 1983, p. 285). Approximately half of the tasks are as fast as the KLM on trial 3, and all but one are on trial 4.

Table 2: KLM-GOMS Prediction of Subtasks (in seconds)

Subtasks	Operators				
	M	H	P	K	Time
Sub1	3	1	0	33	19.96
Sub2	3	0	0	26	16.27
Sub3	20	0	0	158	101.26
Sub4	4	0	0	27	18.09
Sub5	20	0	0	169	106.43
Sub6	4	0	0	37	22.79
Sub7	39	0	0	194	143.83
Sub8	4	0	0	27	18.09
Sub9	40	0	0	186	141.42
Sub10	4	0	0	27	18.09
Sub11	2	0	0	39	21.03
Sub12	2	0	0	9	6.93
Sub13	4	0	0	24	16.68
Sub14	3	0	0	25	15.80
Operators	152	1	0	981	
Time	205.20	0.40	0.00	461.07	666.67

Statistical Modeling of Performance Change

The data set used in this paper is longitudinal with repeated measurements for each participant and for each subtask over time. To deal with non-independency in measurements, we choose to use a linear mixed effects model. The response variable in the data set is the task completion time.

In our linear mixed effects model, the fixed effect is the practice trials that is represented as days. As random effects, we had intercepts for participants and subtasks, as well as by-participants and by-subtasks random slopes for the effect of learning trials over time. This statistical model is adequate for the question of interest in this paper, investigating whether different subtasks have differential learning rates by participants over practice trials. Subtasks and participants are completely crossed, and the task time was repeatedly measured from each participant.

Results

Figure 3 shows a preliminary plot of the 14 subtasks. It shows different patterns of performance change across four days of practice trials. The red dashed horizontal lines are the KLM predictions. Figure 3 suggests the practice trials for four consecutive days allow participants to approximately reach an KLM expert performance except for subtasks 7 and 9. Where the KLM predictions seem to be higher than true experts will be, these subtasks have the largest number of mental and keystroke operators (refer to Table 2). This result casts a question as to whether the number of mental operators are over predicted.

We used the *lme4* package (Bates, Mächler, Bolker, & Walker, 2014) in R to conduct a linear mixed effects analysis of the relationship between the response variable and the covariate predictors including fixed and random effects.

We checked the normality assumption of the data. The Q-Q plot of residuals shows that the residuals are not normally distributed. To address this issue, we performed log-transformation of the data. Our linear mixed effects model then meets the assumption of normality of residuals.

To assess the significance of practice trials (day) as a predictor, we looked at the t-value of the fixed effects. The t-value of the slope estimate is large enough. Thus, we can estimate that the predictor is significant since our dataset is fairly large with 1680 observations.

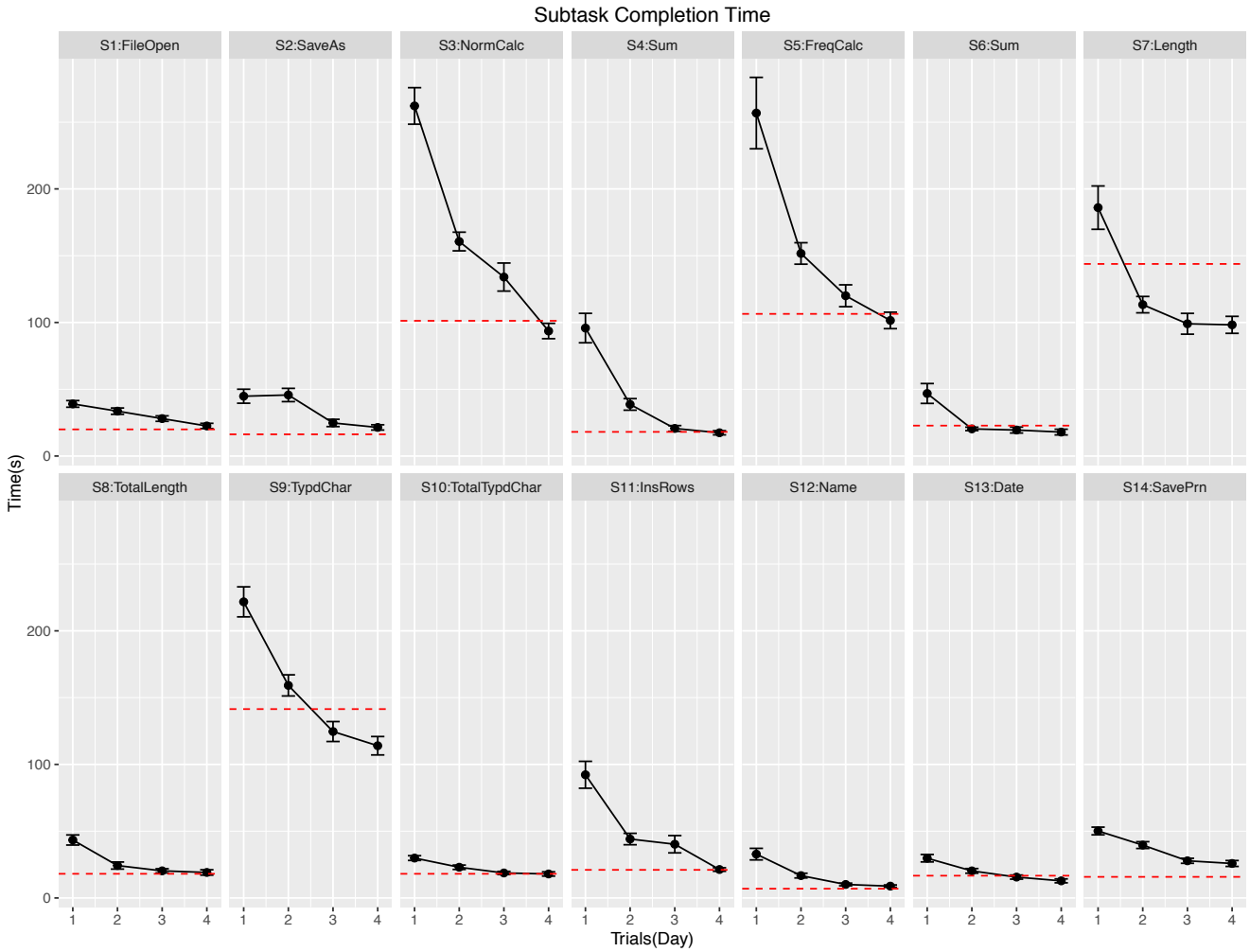


Figure 3: Average subtask completion times (*N*=30) in seconds with mean (solid black) and SEM (as error bars) for each subtask. The red dashed lines are the KLM predictions for each subtask.

We plotted the data to depict all the task completion times over practice trials by the 14 subtasks (N=30), and a linear regression line for each subtask in a log-log coordinates, as shown in Figure 4. There were 48 missing values from 1680 data points (2.9%), but it can be considered that those missing values are acceptable for our model due to the total number of data points.

Besides the fixed effect of practice trials over days, it is of interest to determine how the subtasks differ. We compared two models: one model is a random intercept model both for participants and for subtasks, and the other model is also a random intercept model that has only different intercepts of participants (i.e., without random intercepts of subtasks). The random deviations (residuals, SD=0.17) from the predicted values that are not caused by both subtasks and participants increased in the case of the random deviations (SD = 0.40) only due to participants. This indicates that the subtasks have an effect on the performance change. By performing ANOVA to compare those two models, we can conclude that there is a statistical significance of the subtask effect, $\chi^2(1) = 2681.4$, $p < 0.001$.

As seen in Figure 4, there exist varying slopes by subtasks, indicating different learning rates by each subtask. With regard to the varying slopes of the subtask effect, we compared the model with random intercepts to an alternative model with random slopes for the subtask. We found there are significant differences in learning rates by the random effect of the 14 subtasks, $\chi^2(2) = 115.59$, $p < 0.001$.

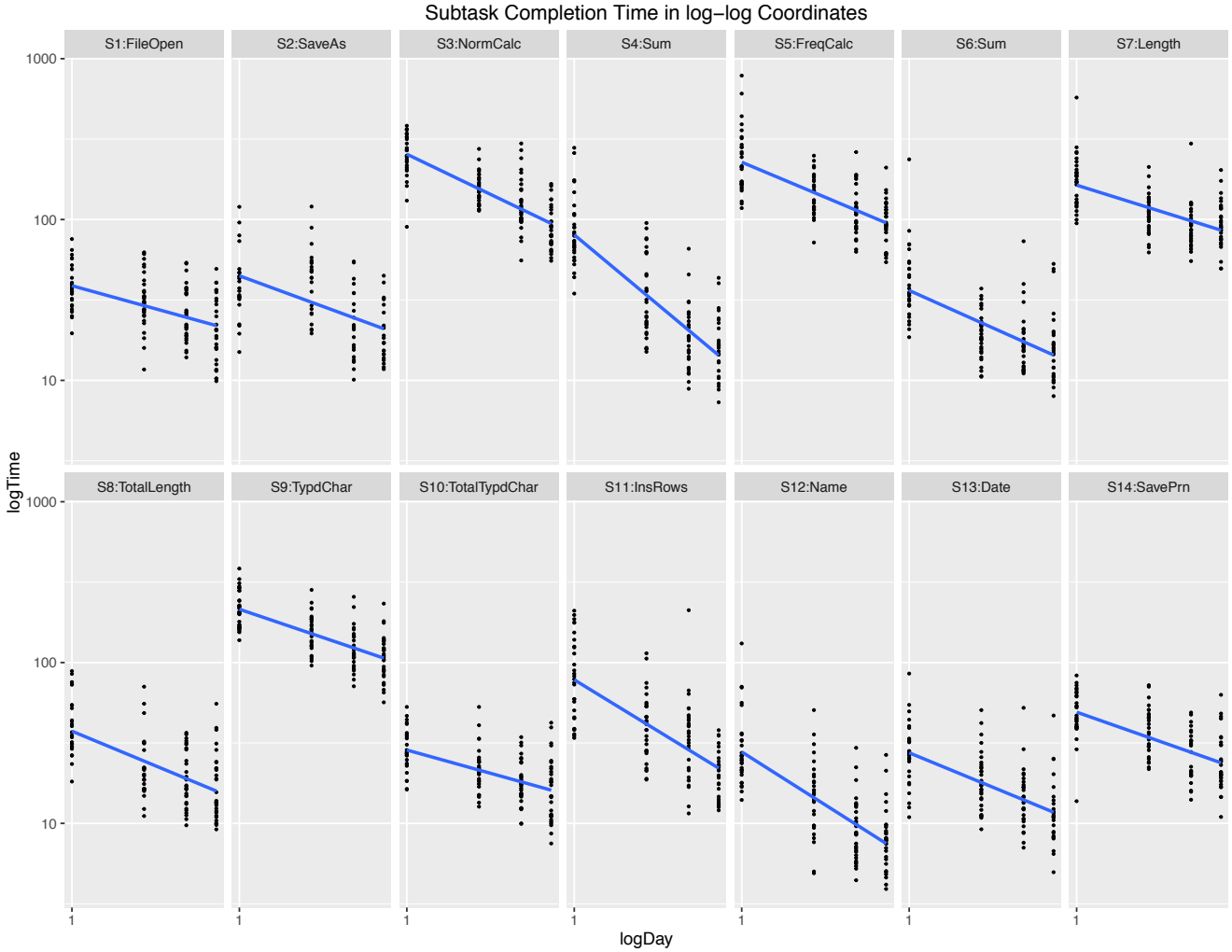


Figure 4. Regression lines with scatter plots for each subtask in a log-log scale.

Discussion and Conclusions

Figure 5 shows differences and similarities in the slope for the predicted time by each subtask. Similar slopes are observed in the subtasks 1, 2, 7, 9, 14, subtasks 3, 5, 6, and subtasks 11, 12. As noted in Table 1, the participants retrieve each keystroke command for the corresponding subtask, such as the unique key commands, C-x C-f, for "Open File", and C-x C-w, for "Save As". In this manner, the operators required for subtask 3 and 5, which are normalization and frequency calculations, are nearly identical. The slopes for the task time predictions are similar

as well. However, it is apparent that the slope of subtask 3 is steeper than the slope of subtask 1. It is interesting that the number of operators of either type in a subtask, particularly when there are fewer than 50, is not correlated with learning slopes.

With regard to the subtasks 3, 5 (normalization and frequency calculations), and subtasks 7, 9 (calculating length and typed characters), those subtasks require a large number of keystroke operators in the spreadsheet subtask. However, the number of keystroke operators might not be what influenced learning because there are other subtasks with steeper slopes and fewer keystroke operators. On the other hand, the keystroke skills are learned for four consecutive days of practice. All these subtasks required participants to repeat 10 calculations per practice trial. This can be considered as motor skill practice with a massed training regimen.

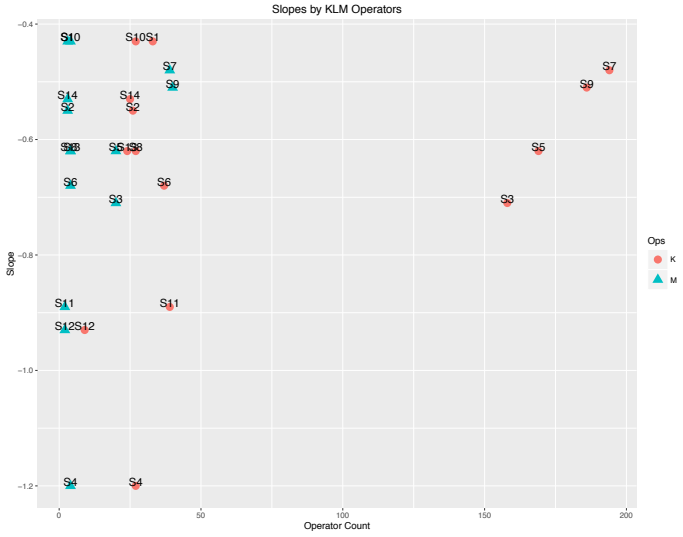


Figure 5: Scatterplot of the varying slopes against operators (Keystroke and Mental). (Lower is greater learning.)

Figure 5 suggests that as mental operators go up, the learning rate goes down, but this seems curious. Regarding mental operators, some subtasks require participants to retrieve a unique keystroke command, and this can lead to higher learning rates. Perhaps these have different effects on learning. For example, to insert two rows, a participant needs to retrieve a declarative memory element, C-u 2 i r (the subtask 11). We can consider that the subtasks 11, 12, and 4 would lead to higher learning rates due to a weak activation of the corresponding element. This notion emphasizes the importance of moving the declarative memory elements to the procedural stage (Fig 1).

This analysis shows that the subtasks vary in learning. We are now analyzing why learning varies so much across subtasks and will be investigating using our existing ACT-R models.

Acknowledgments

This research was supported by grants from the Division of Human Performance Training, & Education at ONR (W911QY-07-01-0004, N00014-10-1-0410, and N00014-15-1-2275). Ysabelle Coutu provided useful inputs.

References

Agre, P., & Shrager, J. (1990). Routine evolution as the microgenetic basis of skill acquisition. In *Proceedings of the 12th Annual Conference of Cognitive Science Society* (pp. 694-701). Cambridge, MA.

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review, 89*(4), 369-406.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York, NY: Oxford University Press.

Anderson, J. R., Fincham, J. M., & Douglass, S. (1999). Practice and retention: A unifying analysis. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 25*(5), 1120-1136.

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2014). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software, 67*(1), 1-48.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum.

Fitts, P. M. (1964). Perceptual-motor skill learning. In A. W. Melton (Ed.), *Categories of human learning* (pp. 243-285). New York: Academic Press.

Heathcote, A., Brown, S., & Mewhort, D. J. K. (2000). The power law repealed: The case for an exponential law of practicie. *Psychonomic Bulletin & Review, 7*(2), 185-207.

Kim, J. W., & Ritter, F. E. (2015). Learning, forgetting, and relearning for keystroke- and mouse-driven tasks: Relearning is important. *Human-Computer Interaction, 30*(1), 1-33.

Kukreja, U., Stevenson, W. E., & Ritter, F. E. (2006). RUI: Recording user input from interfaces under Window and Mac OS X. *Behavior Research Methods, 38*(4), 656-659.

Moon, J. M., & Fu, W.-T. (2008). A situated cognitive model of the routine evolution of skills. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (pp. 935-939). Sage.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 1-55). Hillsdale, NJ: Lawrence Erlbaum.

Paik, J., Kim, J. W., Ritter, F., & Reitter, D. (2015). Predicting user performance and learning in human-computer interaction with the Herbal compiler. *ACM Transactions on Computer-Human Interaction, 22*(5), 1-26.

Rasmussen, J. (1986). *Information processing and human-machine interaction: An approach to cognitive engineering*. New York: Elsevier.

Ritter, F. E., Baxter, G. D., Kim, J. W., & Srinivasmurthy, S. (2013). Learning and retention. In J. D. Lee & A. Kirlik (Eds.), *The Oxford handbook of cognitive engineering* (pp. 125-142). New York, NY: Oxford University Press.

Ritter, F. E., & Larkin, J. H. (1994). Developing process models as summaries of HCI action sequences. *Human-Computer Interaction, 9*, 345-383.

Ritter, F. E., & Wood, A. B. (2005). Dismal: A spreadsheet for sequential data analysis and HCI experimentation. *Behavior Research Methods, 37*(1), 71-81.

Seibel, R. (1963). Discrimination reaction time for a 1,023-alternative task. *Journal of Experimental Psychology, 66*(3), 215-226.

Siegler, R. S. (2006). Microgenetic analyses of learning. In *Handbook of child psychology* (pp. 464-510): Wiley.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

A Metacognitive Agent for Training Negotiation Skills

Christopher A. Stevens (c.a.stevens@rug.nl), Harmen de Weerd (h.a.de.weerd@rug.nl), Fokie Cnossen (f.cnossen@rug.nl), Niels A. Taatgen (n.a.taatgen@rug.nl)
Department of Artificial Intelligence, Nijenborgh 9
9747AG Groningen, The Netherlands

Abstract

Training negotiators remains a difficult and expensive proposition. Negotiators require complex cognitive skills such as theory-of-mind to be successful, but these skills can be difficult to train and measure. Here we present an agent designed to model theory-of-mind for learners and serve as a practice partner for complex negotiations. This agent employs instance-based learning to make decisions about its own actions and to reflect on the behavior of the opponent. This reflection process is used to provide a source of explicit feedback on the opponent's strategy and behavior. In this paper we present evidence that the model is a plausible opponent for students learning negotiation. It is expected that practicing with this agent will improve theory-of-mind abilities in learners and, in turn, improve negotiation performance.

Keywords: Metacognition, Negotiation, Theory-of-mind, Autonomous Agents

Training Negotiation with Artificial Agents

Negotiation is a complex human activity that permeates many aspects of business, politics, and even daily life. There is a myriad of possible negotiation settings, and in most of these settings, there are many possible outcomes. Training negotiators is difficult because the optimal strategy often depends both on the negotiation setting and on the strategy used by one's negotiation partner (Fisher & Ury, 1981; Raiffa, 1982). For instance, an aggressive, unyielding strategy (a.k.a. the "Boulware" strategy (Cross, 1977)) may work very well when the partner is agreeable or flexible. However, the same strategy will lead to stalemate against an aggressive opponent and could harm the potential for future negotiations with fair-minded opponents (Tinsley, O'Connor, & Sullivan, 2002) Therefore, a good negotiator may benefit from using theory-of-mind to infer an opponent's preferences, diagnose an opponent's strategy, and select an appropriate counter strategy. One promising emerging training technique is practicing with artificial agents. Although few studies exist that explicitly evaluate artificial agents as training partners, there is evidence that training with an artificial agent is at least as good as training with a human (Lin, Gal, Kraus, & Mazliah, 2014). In this paper, we present a novel cognitive agent designed to imitate the strategies and theory-of-mind capabilities of human negotiators. Further, we present results from a small-scale pilot study that suggest the agent is capable of performing similarly to humans in a multi-issue bargaining scenario.

Simulation and behavioral studies have shown that theory-of-mind can improve negotiation outcomes. Specifically, agents with more complex theory-of-mind can achieve greater individual and collective outcomes than those with less complex or no theory-of-mind (de Weerd, Verbrugge, & Verheij, 2015). Moreover, negotiating with an agent that has theory-of-mind can encourage humans to use more complex theory-of-mind (de Weerd, Broers, & Verbrugge, 2015). For this reason, we developed an agent that explicitly reasons about the preferences and strategies of its opponent. We expect that practicing with an agent that has these capabilities will provide better learning outcomes than practicing with an agent without theory-of-mind.

The Metacognitive Agent

Our negotiation agent is based on ACT-R's declarative memory system (Anderson, 2007) and instance-based learning (IBL) theory (Gonzalez & Lebiere, 2005). Instance-based learning was chosen as the reasoning mechanism because it provides a flexible method to reason about novel situations based on examples from previous interactions. Moreover, we believe that theory-of-mind in this domain requires explicit, declarative reasoning rather than procedural knowledge. Many negotiation contexts are relatively novel and our participants were not experienced in professional negotiation, so it seems unlikely that people in our target population have sufficient practice to develop a comprehensive set of production rules that match each possible situation.

The agent's memory contains a set of examples (instances) that represent possible negotiation moves and possible contexts in which they may sensibly be used. Each instance is associated with a particular strategy (cooperative or aggressive). For example, when there is a deadlock, a player using a cooperative strategy might concede on a minor issue to break it. However an aggressive player might threaten to quit if his/her offer is not accepted. The agent uses this same knowledge to choose its own moves, to evaluate the player's strategy, and to make inferences about the player's preferences.

The Smoking Ban Negotiation Scenario

The specific setting we consider is a multi-issue bargaining scenario in which a representative of a city council and a representative of small business owners negotiate over the implementation of new anti-smoking regulations. The negotiation involves four issues, each with four or five different options. The task of the negotiators is to reach an agreement which assigns exactly one option to each issue. Despite this simple setup, this setting allows for 400 different possible negotiation outcomes in addition to the opt-out outcome. It also allows a rich set of possible negotiation moves (see Table 1 for definitions of possible moves).

In our setup, each negotiator has preferences that assign a value to each possible option. Higher values are associated with more preferable options. The value of a negotiated out-

come is calculated as the sum of the values of the agreed-upon options. A negotiator therefore aims for a negotiated outcome that his preferences assign as high a value as possible. Importantly, preferences are private information. That is, each player knows their own preferences, but not the preferences of the other.

We pilot-tested this scenario to get a sense of the possible negotiation moves and strategies used by human negotiators. In this pilot test, four pairs of human participants negotiated an agreement for nine different problems with unique preference values. From this pilot testing, we made two observations. Firstly, the people in our sample did not always find optimal agreements and sometimes even accepted negative deals. Our measure of optimality is Pareto optimality. An agreement is Pareto optimal if it there are no possible alternative agreements that could raise one player's score without reducing the other player's score. The human dyads found Pareto optimal agreements in 61% of the problems. Moreover, 21% of the negotiated deals resulted in a negative score for at least one of the negotiation partners. These results confirm that our negotiation setting is sufficiently challenging for a training intervention.

Our second observation is that unlike automated negotiation agents, which submit offers as binding commitments, human players often make offers with lower levels of commitment. In other words, players can discuss preferences on various options without being bound to accept those options later in the negotiation. To capture this, we designed our artificial agent so that it can understand moves with differing levels of commitment. For example, exchanging preferences on an issue implies a low level of commitment, but accepting an offer implies a higher level of commitment.

Instance-based Decision-making

Instance-based learning was implemented here using a modified version of Java ACT-R (Salvucci, 2013). An instance is a set of slot-value pairs that represents a context, an action, and a utility value for that action (Gonzalez & Lebiere, 2005). Table 2 contains the specific slots used in the instances in our metacognitive agent.

To select a move, the model uses the current negotiation context to retrieve an instance from memory. The instances are retrieved using ACT-R's partial matching mechanism (Anderson, 2007). The more similar the instance is to the current context, the more active the instance will be and therefore the more likely the instance will be retrieved. Each instance also has a base activation level. This base activation is constant across all instances in the agent, save for a very small amount of noise (s=.01). The instances in the current model were written by the modelers to ensure a stable, challenging opponent whose cooperative and aggressive strategies were consistent with those found in the literature (Fisher & Ury, 1981; Raiffa, 1982).

As an example, suppose the agent retrieves an instance like the one in Table 3. In this context, the agent is playing aggressively. The agent has proposed an option that is worth a lot of points (4 is the maximum in this setting), and it believes that this option is bad for its partner (the option would cause the partner to lose points). The agent's partner has proposed an option that is worth 3 points to the agent, which is still a very good value for the agent, and it happens to be the agent's next best option. Moreover, the agent is already doing well in the negotiation, because on the other resolved issues, it has already gained 3 points. In this case, if the agent retrieves this instance, it will try to pressure its opponent to take a loss even though a more mutually beneficial option is probably available. This is indicated by the Insist move. By contrast, a cooperative instance would likely accept the opponent's bid in this situation.

Theory-of-mind

Negotiators strike a delicate balance between cooperation and competition (Lax & Sebenius, 1986). Cooperation helps encourage agreement and trust between negotiators but it can be exploited by competitive negotiators. A good negotiator is mindful of this and takes steps to prevent exploitation. One way to achieve this is to use theory-of-mind to infer the opponent's strategy. Our agent achieves this by taking the perspective of its partner and using its own knowledge to evaluate the partner's strategy. The agent then attempts to match the toughness level of the opponent. If the partner is cooperative, the agent will also be cooperative. But if the partner is aggressive, the agent will become more aggressive. This meta-strategy has been observed in humans in negotiation and coordination games (Kelley & Stahelski, 1970; Smith, Pruitt, & Carnevale, 1982) and has been shown to be effective at encouraging cooperation in the Prisoner's Dilemma (Stevens, Taatgen, & Cnossen, 2016).

Each time the user makes a negotiation move, the agent assumes the perspective of the user and uses its own instances and decision processes to infer the user's strategy and beliefs. Of course, the agent does not have access to the same information when interpreting the opponent's actions as it does when it is choosing its own (e.g. exact preference values, user's chosen strategy). In these cases, the agent fills in its best guess or leaves the slot empty. Fortunately, ACT-R's declarative memory system is robust to missing information and memory retrievals can be made without specifying all of the slots. As more memory retrievals are made, the agent updates its best guess about the user's strategy. The agent evaluates both the user's reaction to the agent's move (if applicable) and the countermove made by the user. This results in up to two memory retrievals per negotiation turn.

The agent's memory holds three sets of preference values: the agent's own preferences, the agent's beliefs about the user's preferences, and the agent's beliefs about the user's beliefs about the agent's preference values. At the beginning of the negotiation, the agent has no beliefs about its opponent's preferences or beliefs. As the negotiation progresses, the agent gradually adds information to these sets based on the information found in instances retrieved during theory-of-mind. For example, suppose the agent retrieves the in-

Table 1: Overview of possible negotiation moves.

Move	Explanation
Invite	Elicit an offer from the trading partner on at most one issue. *Example:* "What would you like for the scope of the smoking ban?"
Inform	Inform the trading partner that the player likes or dislikes a single option of a given issue (can also indicate no preference). Liking an option that the partner has suggested or liked results in an agreement (see Agree) *Example:* "For me, a 10% increase in tobacco taxes would be difficult."
Suggest	Ask the partner to commit to a specific option on 1 or 2 issues. **Subtypes:** ***Concede -*** Suggest an option that you haven't Suggested before; ***Insist -*** Suggest an option that have Suggested before; ***Exchange -*** Suggest one option from two different dimensions. Both options are conditional. If one is rejected, so is the other. *Example:* "Would you agree to all outdoor smoking allowed in exchange for a 25% increase in tobacco taxes?"
Agree	Agree to the most recent Suggest or Inform move of the trading partner. This is a non-binding commitment. *Example:* "We can do it as you suggested."
Finalize	Commit to a given negotiation outcome. If the partner accepts, this commitment is binding. *Example:* "So to summarize, I think we should go for all outdoor smoking allowed, no change in tobacco taxes, anti-smoking television advertisements, and a ban on tobacco vending machines"
Accept	Accept the most recent Finalize move of the trading partner. This is a binding commitment. *Example:* "That's a deal."
Withdraw	Causes immediate negotiation failure. *Example:* "We cannot seem to reach agreement. Let's stop negotiating."
Final Offer	Commit to a given negotiation outcome and force the trading partner to either accept this outcome or withdraw from negotiation. This is a binding commitment. The negotiator cannot resume the negotiation if the partner rejects the offer. *Example:* "I think we should settle on all outdoor smoking allowed, no change in tobacco taxes, anti-smoking television advertisements, and a ban on tobacco vending machines. This is my final offer."

stance in Table 3 to interpret its partner's move. According to the instance, one situation in which a player might insist is when they have a strong preference for their current bid and a negative preference for the opponent's bid. Therefore, the agent guesses that the opponent's preference for the their bid is 4.0 and the opponent's preference for the agent's bid is -2.0. These two values are then used to retrieve instances in later turns by filling in the "my-bid-value-me" and "opp-bid-value-me" slots respectively.

In a similar way, when the agent submits a move, the agent makes a guess about how its move influences its opponent's beliefs. For example, suppose the agent indicates that it likes a particular option ("Positive Inform"). The instance retrieved by the agent indicates that the this move is appropriate when the agent has a positive preference for the option (e.g. the value in the "next-bid-value-me" slot is 2.0). Now, the agent will believe that its partner believes that it has a preference of 2.0 for the option. This belief is then used to fill in slots during the agent's theory-of-mind reasoning process.

The process of inferring the user's strategy is similar to that of inferring preferences. As each new instance is retrieved, the agent notes whether the instance is cooperative, aggressive, or neutral. When the instances are cooperative or aggressive, the agent becomes more confident that the user is using that strategy. The confidence value for a given strategy is the activation level of that strategy in memory divided by the total activation of all strategies in memory. When this value exceeds a certain threshold, the model will switch to the appropriate counter-strategy. The threshold is a free parameter of the agent, and can be changed depending on the negotiation context. By default it is set to 0.55.

The agent adjusts its strategy according to the perceived aggression of the opponent. The agent has three modes: cautious, cooperative, and aggressive. The agent begins in cautious mode. This mode is designed to encourage cooperation from the opponent while still guarding against aggression. In this mode, the model prefers neutral moves, followed by cooperative, and then aggressive. If the agent believes the opponent is behaving cooperatively, it will enter cooperative mode, in which the agent favors cooperative moves, followed by neutral, and then aggressive. Finally, if the agent is confident that the opponent is unconditionally aggressive, then it will switch to aggressive mode, in which it favors aggressive moves, followed by neutral, and then cooperative.

Table 2: Structure of an instance in the metacognitive agent

Move type	Explanation
Strategy	The strategy associated with the instance
My-bid-value-me	The number of points the agent's bid is worth to the agent.
My-bid-value-opp	The number of points that the agent believes its bid is worth to the user.
Opp-bid-value-me	The number of points the user's bid is worth to the agent.
Opp-bid-greater	True if the user's bid is at least as much as the agent's current bid, False otherwise.
Next-bid-value-me	The number of points that the next best option is worth. The next best option is defined as the option closest in value to the current one (Not including those that are worth more than the current option.)
Overall-value	The total value of all options that have been agreed upon so far. This is a measure of how the negotiation is going. If it is negative, negotiation is likely to result in an unacceptable outcome.
My-move	The move that the agent should take in this context.

Table 3: An example of an aggressive instance.

Slot name	Value
Strategy	Aggressive
My-bid-value-me	4.0
My-bid-value-opp	-2.0
Opp-bid-value-me	3.0
Opp-bid-greater	false
Next-bid-value-me	3.0
Overall-value	3.0
My-move	Insist

Graphical Interface

Learners can interact with the agent through a graphical interface (see Figure 1). The interface contains five zones. The top zone shows the state of the negotiation. This includes a representation of the negotiation agent's current cooperativeness and a transcript of the negotiation. In this transcript, simulated negotiation dialog is shown in green and simulated agent dialog is shown in orange. This simulated dialog is taken from transcripts of human-human dyads participating in an earlier pilot study on the smoking ban scenario.

The second zone shows the possible ways in which the learner can respond to an offer made by the agent (if any). If the agent has made an offer, the learner may give it a positive, negative, or neutral evaluation. A positive evaluation indicates a tentative agreement, a negative evaluation indicates that an offer is undesirable, and a neutral evaluation states that the offer is under consideration. If the agent has not made an offer, this zone is disabled.

The third zone features the possible actions that can be performed by the user. Actions that are impossible to take at the moment are disabled. The third zone shows the four issues, each with its own options. The background color of each option indicates the evaluation of the option for the user. Darker red colors indicate options that are increasingly more negative, while darker blue options indicate increasingly more positive options. In addition, for each issue, colored triangles indicate the option most recently offered by the user (green) and the negotiation agent (orange). The final zone of the interface gives a preview of the move the user is about to make, and a button to submit that move.

The action selected in the third zone determines what can be selected in the fourth zone. To help the user, actions are grouped by their level of commitment. In addition, two separate buttons are used for proposing and exchanging offers. Proposing offers are offers that assign a single option to exactly one issue, while exchanging offers are offers that assign a single option to exactly two issues. Note that an exchanging offer is interpreted as a temporary offer. If an exchanging offer is not accepted, the triangles indicating the most recently offered option revert to their previous positions. The interface is of course more restrictive than a real-life negotiation. For example, the interface does not allow users to make offers on more than two separate issues. In addition, the interface automatically handles proper Agree, Accept, Finalize, and Final Offer moves. This means that a user can only make an Agree move when the negotiation agent has made an offer and Accept when the agent has made a Finalize or Final Offer move. A user can only make a Finalize or Final Offer move if a green triangle indicates an option for each issue. This means that users cannot attempt to Agree to offers that have not been made, or make partial Finalize moves.

Agent Feedback

During the negotiation, the agent accumulates data about the players' actions to present as feedback. The agent provides feedback on two different aspects of the learner's performance: negotiation style and outcomes. Negotiation style concerns the learner's strategy (cooperative or aggressive) and outcomes refers to the utility of the agreement reached for both players.

Throughout the negotiation, the metacognitive agent evaluates its trading partner on his/her negotiation style. After each action, feedback on the perceived cooperativeness of the action, the agent's beliefs about the preferences of the player, and the accumulated perception of the cooperativeness of the agent's trading partner is available immediately to display as feedback. This includes changing the facial expression of the agent to happy (cooperative), angry (aggressive), or neutral

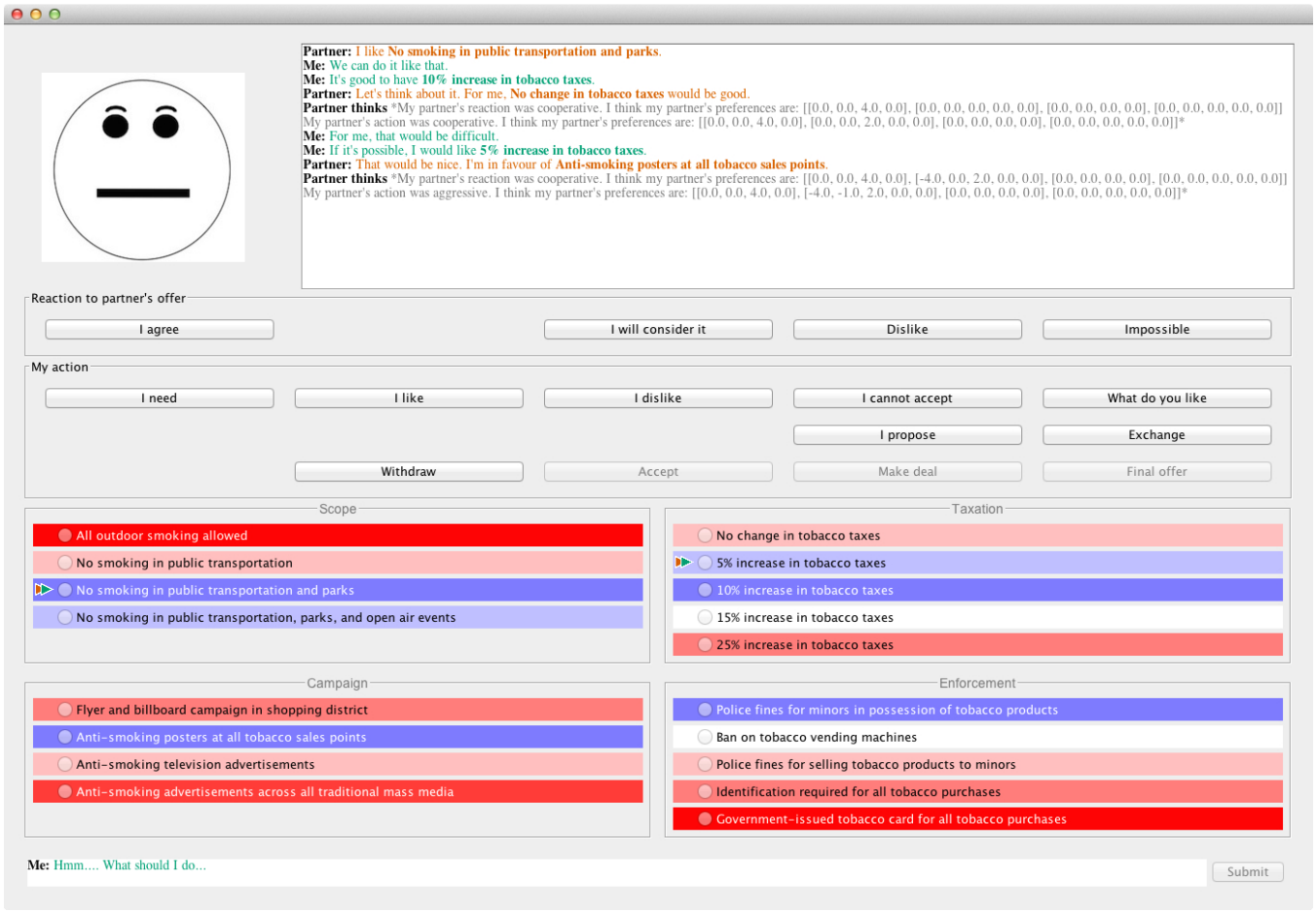


Figure 1: The graphical interface of the agent.

via a line-drawing of a smiley face. Moreover, a line of text is added to the transcript showing whether the agent believes the move was cooperative or aggressive. This line also includes a representation of the agent's beliefs about the user's preferences (see Figure 1).

Once a negotiation has finished and the outcome is known, additional feedback is available about the economic outcome. First, the GUI informs the player how many points he/she gained or lost as a result of the deal. These points are based directly on the colors of the preference panel in the GUI. A good negotiator does not accept a deal that is worse than no deal (i.e. has a negative score), and the agent will inform the learner when this occurs. In addition to the individual outcome, the agent evaluates the Pareto optimality of the agreement and displays it in the GUI.

Pilot experiment

To test the agent's ability to negotiate rationally with human users, we conducted a pilot study. In this study, six human users negotiated with the agent on the same nine problems we used in the pilot experiment with human dyads. The agent always represented the business side. Overall, the results of the pilot study are encouraging, and suggest that the agent is a competent negotiation partner. The agent did not exploit, nor was it exploited by the human users. The agent earned an average of 32.2 (out of 88) possible points while the human users scored an average of 33.7 (out of 86) points. Thus, agreements were similarly beneficial for both partners. The human-human dyads also achieved a relatively even point distribution, albeit with a slight advantage for the business side. Moreover, we compared the agent-human dyads to the human-human dyads on rate of agreement, ability to find Pareto optimal outcomes, and acceptance of negative outcomes (see Table 4). We find that the humans reach a similar number of agreements when negotiating with the agent as when negotiating with each other. Furthermore, the human-agent dyads find a similar (though slightly lower) percentage of Pareto optimal deals. Finally, the human-agent dyads reached fewer negative deals than the human-human dyads. This does not necessarily mean that the human-agent dyads were superior at avoiding negative deals. It is possible that the Withdraw option was more salient in the human-agent dyads due to the GUI. Also, in the human-agent case, the human users received explicit feedback about their scores after every round. Therefore it was clearer how the colors mapped onto overall scores.

Table 4: Comparison of human-human dyads to human-agent dyads

	Human-Human (n = 4)	Human-Agent (n = 6)
Mean business score	33 (20)	32 (3)
Mean council score	29 (7)	34 (8)
% Agreement	78	74
% Pareto optimal	61	52
% Negative deals	21	2

Note. In the human-agent dyads, the business side was always played by the agent. All percentages represent percentage of the total number of trials. SD's are presented in parentheses.

Future Directions

The present pilot study of course does not test the educational outcomes resulting from training with the agent. Evaluation of learning gains is ongoing. In future studies, we aim to test the extent to which training with the agent improves outcomes not only in the smoking ban scenario, but also in other negotiation contexts.

Currently, the agent possesses instances that were hand-coded by the authors and the base activation level does not change. However, allowing the agent to learn the utilities of the instances could result in a more dynamic, and potentially more intelligent, opponent. This is possible, but challenging, for a task like negotiation. Instance-based learning requires a measure of utility, and the utility of a particular negotiation move is not always immediately clear. Therefore, implementing learning in such an agent requires careful consideration of the learning and social context of the negotiation to avoid chaotic agent behavior.

This agent is not designed to be restricted to a point-and-click interface. Rather, it is meant to be a component of a larger system known as Metalogue, a large, multimodal negotiation trainer capable of simulating a real negotiation dialogue (Helvert, Rosmalen, Borner, Petukhova, & Alexandersson, 2015). In the coming months, the agent will be incorporated into this system, and will function as a decision engine. As it does in the GUI setting, the agent will play the role of a negotiation partner and trainer discussing the options of a smoking ban. However, in this case, the learner will be able to interact with the model through speech rather than through clicking buttons. Moreover, the agent will be portrayed by a virtual avatar with speech and gestures of its own.

Summary

Here we have presented a novel cognitive agent that reasons about the goals and strategies of human partners to successfully engage in a negotiation task. This agent leverages established cognitive theories, namely ACT-R and instance-based learning, to generate plausible, flexible behavior in this complex setting. Our preliminary results suggest that our cognitive agent could play a role in training effective negotiators.

Acknowledgments

This work was funded by European Union Grant 611073: Multiperspective Multimodal Dialogue (METALOGUE).

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press.

Cross, J. G. (1977). Negotiation as a learning process. *Journal of Conflict Resolution*, *21*(4), 581–606.

de Weerd, H., Broers, E., & Verbrugge, R. (2015). Savvy software agents can encourage the use of second-order theory of mind by negotiators. In *Proceedings of the 37th annual conference of the cognitive science society.* (pp. 542–547). Pasedena, California.

de Weerd, H., Verbrugge, R., & Verheij, B. (2015). Negotiating with other minds: the role of recursive theory of mind in negotiation with incomplete information. *Autonomous Agents and Multi-Agent Systems*.

Fisher, R., & Ury, W. L. (1981). *Getting to Yes: Negotiating Agreement Without Giving In*. London: Penguin Group.

Gonzalez, C., & Lebiere, C. (2005). Instance-based cognitive models of decision-making. In D. Zizzo & A. Courakis (Eds.), *Transfer of knowledge in economic decision making.* New York: Palgrave McMillan.

Helvert, J. v., Rosmalen, P. V., Borner, D., Petukhova, V., & Alexandersson, J. (2015). Observing, coaching and reflecting: A multi-modal natural language-based dialogue system in a learning context. In *Proceedings of the 11th international conference on intelligent environments* (p. 220).

Kelley, H. H., & Stahelski, A. J. (1970). Social interaction basis of cooperators' and competitors' beliefs about others. *Journal of Personality and Social Psychology*, *16*(1), 66–91.

Lax, D., & Sebenius, J. (1986). *The Manager as Negotiator*. Boston: Harvard Business School Press.

Lin, R., Gal, Y., Kraus, S., & Mazliah, Y. (2014). Training with automated agents improves people's behavior in negotiation and coordination tasks. *Decision Support Systems*, *60*(1), 1–9.

Raiffa, H. (1982). *The art and science of negotiation*. Cambridge: Belknap Press.

Salvucci, D. D. (2013). Integration and reuse in cognitive skill acquisition. *Cognitive Science*, *37*(5), 829–860.

Smith, D. L., Pruitt, D. G., & Carnevale, P. J. (1982). Matching and mismatching: The effect of own limit, other's toughness, and time pressure on concession rate in negotiation. *Journal of Personality and Social Psychology*, *42*(5), 876–883.

Stevens, C. A., Taatgen, N., & Cnossen, F. (2016). Instance-based models of metacognition in the prisoner's dilemma. *Topics in Cognitive Science*, *8*(1), 322–334.

Tinsley, C. H., O'Connor, K. M., & Sullivan, B. a. (2002). Tough guys finish last: The perils of a distributive reputation. *Organizational Behavior and Human Decision Processes*, *88*(2), 621–642.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Cognitive Models of Prediction as Decision Aids

Christian Lebiere, Don Morrison (cl@cmu.edu, dfm2@cmu.edu)
Department of Psychology, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213 USA

Tarek Abdelzaher, Shaohan Hu (zaher@illinois.edu, shu7@illinois.edu)
Department of Computer Science, University of Illinois at Urbana Champaign
201 N Goodwin Ave, Urbana, IL 61801 USA

Cleotilde Gonzalez (coty@cmu.edu)
Department of Social and Decision Sciences, Carnegie Mellon University, 5000 Forbes Avenue
Pittsburgh, PA 15213 USA

Norbou Buchler,Vladislav D. Veksler (norbou.buchler.civ@mail.mil, vladislav.d.veksler.ctr@mail.mil)
Cognitive Sciences Branch, Human Resources and Effectiveness Directorate, U.S. Army Research Laboratory
Aberdeen Proving Ground, MD 21005 USA

Abstract

We consider the use of cognitive models as both models of human cognitive function and human-compatible decision aids. The domain of application is prediction based on partial information in the context of emergency events where the availability and timeliness of information is limited. The cognitive model is based on the memory retrieval processes of the ACT-R cognitive architecture, most specifically its underpinnings in the rational analysis of cognition. The model is shown to capture well the temporal and spatial characteristics of the data. Finally, we discuss potential issues in the application of cognitive models as decision aids and recommender systems, in particular the ability to introspect in the workings of the model to select data most suitable for the human decision making process.

Keywords: Rational Analysis, Cognitive Architecture, Long-Term Memory, Decision Making, Decision Aids, Recommender Systems.

Introduction

In the Age of Big Data, we are confronted with an increasingly rich and rapid flow of information. While the availability of data is increasing seemingly exponentially in our personal and professional lives, our basic human capabilities are not keeping pace. Recognizing with his customary foresight the increasingly deep disconnect between our abilities and the demands placed upon them, Herbert A. Simon once said, "Moore's Law fixes everything but us".[1]

Of course, technology has the potential to be the solution as well as the problem. Adaptive information retrieval tools such as search engines are helping us access and filter vast and diverse knowledge resources. In a more proactive way, personal electronic assistants such as Siri and Google Now offer to manage our data flows and provide us with timely, contextual information.

However, interpreting information and using it to make decisions is considerably more complex than simply making it available. Decision aids, including recommender systems, have been proposed to assist and delegate complex human decision-making. Leveraging the Big Data wave itself, those systems are typically data-driven, exploiting statistical regularities to extrapolate to similar situations. For instance, Netflix recently organized a competition to develop better algorithms for recommending movies by relying on ratings of viewers with similar tastes to a given customer. A fundamental problem with this approach is its opaque nature. When it fails, it tends to do so in ways unexpected and incomprehensible to human users, undermining trust in a system that not only performs poorly but cannot explain its own failures.

One potential solution is to design personal assistants that work in ways similar to humans, making them both more transparent and more compatible. Recently, a number of proposals have been made to measure artificial intelligence in more effective ways than the classic Turing Test, in particular by having it perform more typical tasks in human-like ways (AI Magazine, 2016). Going even further, the suggestion has been made to design intelligent agents based on the structure and mechanisms of the human brain (e.g., Stocco et al., 2010). For purposes of decision aids, such biologically inspired cognitive architectures might be a bridge too far. For instance, Google's PageRank algorithm might work in a way roughly similar to human associative memory, but few users would presumably care whether it mimics the structure of the hippocampus or the posterior cortex.

Cognitive architectures and models have primarily been developed as computational instantiations of theories of cognition. For purposes of serving as decision aids to human users, it is tempting to adopt the traditional AI view of treating them as black boxes and arguing that compatibility

[1] While Moore's Law might finally be running out of steam, it has been replaced by the exponentially increasing availability of massively distributed computation and sensor resources (Kurzweil, 2006).

with the human decision maker is primarily required from a functional, behavioral point of view. In terms of the Marr levels of analysis (Marr & Poggio, 1976), that would mean that what matters is primarily their functionality at the computational level rather than the algorithmic level or the implementational level. We disagree with that view.

Instead, we argue that, while the implementational level might not be directly relevant other than perhaps for purposes of scalability and efficiency, compatibility with the human decision maker at the algorithmic level is essential for truly effective interaction. Computational equivalence only enables a relatively superficial integration of outcomes, while algorithmic equivalence enables a deeper integration of processes.

We illustrate the distinction by introducing two functions of a cognitive model as decision aid: prediction and source selection. Prediction involves generating a recommended decision for the user to follow, and as such only requires computational compatibility. However, it leaves the user with little choice beyond accepting or rejecting the recommendation in its entirety. Source selection consists in selecting a subset of information on which the human user would base his own decision. While this enables richer interaction between user and decision aid, it also requires deeper compatibility, down to the algorithmic level, because the selection process requires integration with the processes of the human decision maker.

In the rest of this paper, we introduce a decision-making task based on a real world data set of emergency situations. We then describe a model of the task based on a rational analysis of cognition, and present quantitative results. Finally, we discuss implications for the design of cognitively inspired decision aids and recommender systems, and point out future work directions.

Task and Data

We focus on the problem of data extrapolation in participatory sensing applications, where users both use and provide information to the system, in the face of disruptive pattern changes, such as those that occur during natural disasters. We consider cases where resource limitations or accessibility constraints prevent attainment of full real-time coverage of the measured data space, hence calling for data extrapolation. Many time-series data extrapolation approaches are based on the assumption that past trends are predictive of future values. These approaches do not do well when disruptive changes occur. An alternative recourse is to consider only spatial correlations. For example, certain city streets tend to get flooded together after heavy rain (e.g., because they are at the same low elevation), and certain blocks tend to run out of power together after a thunderstorm (e.g., because they share the same power lines). Understanding such correlations can thus help infer state at some locations from state at others when disruptive changes (such as a flood or a power outage) occur.

We evaluate our prediction model through a real-world disaster response application. In November 2012, Hurricane Sandy made landfall in New York City. It was the second-costliest hurricane in United States history (surpassed only by hurricane Katrina) and the deadliest in 2012. The hurricane caused widespread shortage of gas, food, and medical supplies as gas stations, pharmacies and (grocery) retail shops were forced to close. The shortage lasted about a month. Recovery efforts were interrupted by subsequent events, hence triggering alternating relapse and recovery patterns. The daily availability of gas, food, and medical supplies was documented by the All Hazard Consortium (AHC), which is a state-sanctioned non-profit organization focused on homeland security, emergency management, and business continuity issues in the mid-Atlantic and northeast regions of the United States. Data traces[2] were collected in order to help identify locations of fuel, food, hotels and pharmacies that may be open in specific geographic areas to support government and/or private sector planning and response activities. The data covered states including WV, VA, PA, NY, NJ, MD, and DC. The information was updated daily (i.e., one observation per day for each gas station, pharmacy, or grocery shop).

With these points of interest sites and input data as ground truth, we evaluate the model predictions. The metrics we use are accuracy of inference and amount of data needed. We break time into daily cycles to coincide with the AHC trace. We then plot the performance of the model when a configurable amount of today's data is available (in addition to all historic data since the beginning of the hurricane).

We evaluate the solutions on November 3rd, and November 8th. November 8th corresponds to a period of disruptive change due to a second snowstorm that hit after Sandy, causing massive temporary relapse of recovery efforts due to new power outages, followed by a quick state restoration to the previous recovery profile. November 3rd is an example of a period of little change, when damage was incurred but recovery efforts have not yet been effective. The same trend was observed for all datasets, namely, gas, pharmacy, and food.

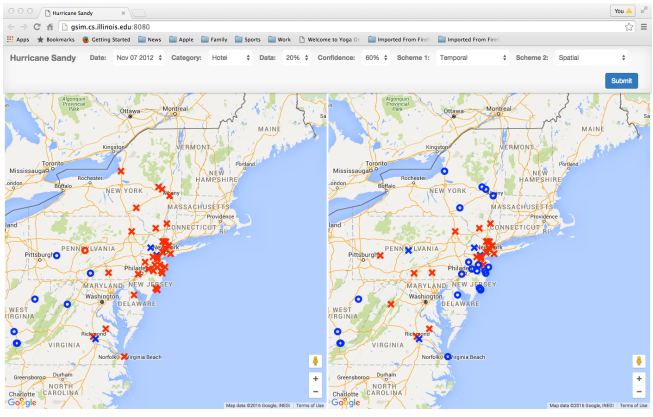


Figure 1: Data Extrapolation Task Interface.

[2] Available at: http://www.ahcusa.org/hurricane-Sandy-assistance.htm

Figure 1 displays a snapshot of the interface that we used to display the model results during model development. While we will focus in this paper on quantitative results for model evaluation, visualizing the spatial and temporal patterns of model prediction helped us understand the workings of the model and its strengths and shortcomings. It also helped us experiment with the model efficiently in exploring parameter settings and comparing model versions. Pull-down menus let the modeler easily select the date of the comparison, the category of data (pharmacy, food, gas), the sampling rate (percentage of the day's data to use in addition to historical data), the confidence threshold (probability to label an outlet as open, defaulting at an unbiased 50%) and any two versions of the model (see next section) to compare side by side against each other. Circles/crosses indicated a prediction that the outlet was open/closed. Color indicated the correctness of the prediction, with blue and red representing correct and incorrect respectively, while grey indicates no prediction was made because that data point was sampled. The data points were plotted on a Google Maps overlay of the geographical area, allowing the modeler to zoom in and out on various areas.

Cognitive Model

While prediction can be viewed as a specialized exercise best left to domain experts and statisticians, e.g., weather forecasting, stock market investing, sports betting, it also forms the implicit basis of many common everyday tasks. Previous models have shown its ubiquity in domains ranging from game playing (West & Lebiere, 2001) to sports (Lebiere et al., 2003), decision-making (Erev et al., 2010), and learning event sequences (Wallach & Lebiere, 2000).

While prediction can require the use of elaborate strategies and expert knowledge, those approaches are highly domain-specific and thus generalize poorly and tell us little about the basic nature of cognition. More fundamentally, complex approaches still seem to rely on a common basis of implicit statistical inference (e.g., Oaksford & Chater, 2007). The rational analysis of cognition (Anderson, 1990) has argued that our cognitive mechanisms have evolved to reflect the statistical structure of the environment. These regularities are quite pervasive and are displayed by our cognitive systems even when they are unwarranted and result in cognitive biases (Lebiere et al.You, 2013).

The rational analysis of cognition can offer a computational-level account of cognitive prediction. To achieve an algorithmic account with constrained quantitative predictions, we used the ACT-R cognitive architecture (Anderson et al., 2004). The mechanisms of its declarative module, in particular, reflect pervasive statistical patterns of the environment such as the power laws of learning and forgetting. As prediction relies on the knowledge of past events, it is logical to base the model on retrieval of information from long-term declarative memory.

In ACT-R, information is represented in declarative memory in the form of chunks, which are structured objects consisting of a set of attributes (also known as slots) with associated values. Chunk complexity (i.e., number of attributes) is typically limited, reflecting capacity constraints such as the size of working memory (Miller, 1956; Cowan, 2001). For instance, it would be unreasonable to store the entire history of an outlet or a whole day's data in a single chunk. Beyond capacity limitations, theories of chunk creation also typically limit their content to information that is available simultaneously at a given point in time and thus can plausibly be bound together in a new chunk structure.

Therefore, each chunk in memory represents the availability of a given outlet on a given day. Attributes that are represented include the identity of the outlet (itself represented as another chunk) and its status: open or closed (also represented as a chunk). The specific day could have been represented as a third attribute, although we decided against it for two reasons. First, it is slightly implausible that people would explicitly label each memory with the date of the day in which it was formed. Second, it would have resulted in a proliferation of memory chunks (one for each day and outlet) without markedly affecting the model predictions when the blending mechanism is used (see below).

Instead, time is represented implicitly in the activation of the corresponding chunk. The base-level activation B_i of a chunk i reflects its history of (re)creation and access as follows:

$$B_i = log \sum_{j=1}^{n} t_j^{-d} \qquad (1)$$

Where t_j is the time lag since the jth occurrence of the chunk, n is the total number of occurrences, and d is the decay rate (typically fixed at 0.5, as is the case in this model). For any given outlet, at most two associated chunks exist in memory: one recording that the outlet is closed and another recording that it is open on a given day. The base-level activation of these chunks will be reinforced with each occurrence of the respective event. The temporal version of the model then obtains a prediction for the status of a given outlet by retrieving the most active chunk associated with that outlet and returning the status stored in that chunk. Because the total activation A_i of chunk i also includes in addition to the base-level activation a stochastic component controlled by noise parameter s (using the typical value of 0.25 here), the retrieval process is probabilistic, described by the probability $P(i)$ following the Boltzmann (softmax) distribution over all candidate chunks j:

$$P(i) = \frac{e^{\frac{A_i}{s}}}{\sum_j e^{\frac{A_j}{s}}} \qquad (2)$$

For a given outlet, only two chunks will compete for retrieval, and the winning chunk will reflect a combination of frequency and recency of the associated outcome, which

is generally the temporal properties that are desired for prediction.

However, as mentioned earlier, temporal criteria are of limited usefulness when facing sudden disruption such as natural disasters. While a given outlet is usually open (frequency), and was open yesterday (recency), it may not be open today if a disaster event happened in the meantime. In that case, spatial factors constitute an additional basis for making predictions. Assuming lack of specific event knowledge, e.g., where the storm happened to hit, the most direct basis for including spatial factors is the limited known availability of nearby outlets. In the absence of additional semantic information (e.g., the outlet brand), the most direct information to use when attempting to generalize across outlets is their spatial location.

Specifically, the spatial component of the model makes use of the partial matching mechanism in memory retrieval, which allows for chunks that do not exactly match the requested pattern to be considered for retrieval, but with a penalty that reflects the degree of mismatch. Specifically, the activation A_i of chunk i is now the sum of the base-level activation and a mismatch penalty term:

$$A_i = B_i + MP * \sum_k Sim(v,d) \quad (3)$$

where *MP* is a mismatch penalty scaling parameter (set in this model at a fairly standard value of 2.0) applied over all k pattern components specified in the retrieval request (only the outlet identity in this case) and *Sim(v,d)* is the similarity penalty between the corresponding value d requested and the actual value v present in the chunk. To avoid introducing needless free parameters, the similarity between outlet chunks is set to a linear function of the geographic distance between them, scaled such that a distance of 25 miles corresponds to a penalty of 1 unit of activation.

When making a prediction for a given outlet, the model will therefore not only consider the history of that given outlet as expressed in the base-level activation of the two associated chunks, but also chunks associated with other outlets as well, with a preference for those closer to the given outlet. Note that unlike that is the target of the prediction, some of those outlets will a known status for the present day, significantly increasing the base-level activation of the corresponding day. Thus the retrieval process will reflect a competition between the recency (and frequency) of outcomes, as reflected in the base-level activation, and its (spatial) relevance, as reflected in the mismatch penalty term.

The final component of the model concerns how to aggregate the relevant knowledge. As specified in the retrieval equation (2), one could simply select the most relevant chunk and return the associated outcome (open or closed). However, that would leave the prediction relying on a comparatively small piece of information, e.g., a chunk of limited relevance being retrieved purely through recency bias or simply the stochasticity of the process. To reflect people's ability to weigh a sizable part of their knowledge base when making predictions (e.g, Lebiere, 1999), the blending retrieval mechanism specifies how to return a value V (in this case, the availability prediction of a specific outlet) that reflects the consensus of the entire set of considered chunks, weighted by their respective probability of retrieval *P(i)*:

$$V = argmin \sum_i P(i) * (Sim(V, V_i))^2 \quad (4)$$

Where *Sim(V, V_i)* is the similarity between the consensus value V and the value V_i proposed by chunk i. In this model, those values returned by the retrieval process are the outlet availability values: open or closed. Treating those values as binary would result in a process where the evidence for each outcome in the form of the activation of the chunks representing that outcome would be weighted against that of the competing outcome, and the greater one selected.

However, a more general decision process is also possible. By setting those values as numerical outcomes (e.g., 1 for open and 0 for closed) and assuming linear similarities in that range (the default, as for distance similarities earlier), the consensus value V will be somewhere in that interval reflecting the degree of preponderance of one outcome over the other. That value can then be interpreted as a confidence value in the open outcome, and assessed against a probability decision threshold (as mentioned in the description of Figure 1). This reflects the requirements of real world applications, e.g., where one might not want to predict that an outlet is open during an emergency without a fairly high certainty. However, we will only consider majority decisions (i.e., probability threshold of 0.5) in the following results section.

Results

In the absence of comparable human data, we examine the prediction performance of the model on a functional basis, but also looking to assess its cognitive plausibility. We also report results for the temporal and spatial versions of the model to assess the relative contribution of the two mechanisms.

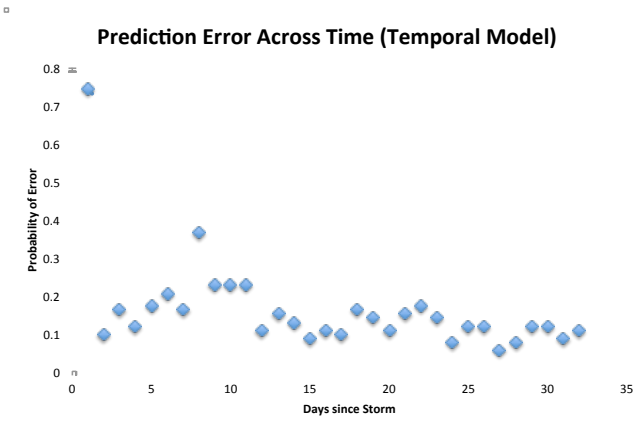


Figure 2: Performance of Temporal Model Across Time.

Figure 2 reports the aggregate performance of the temporal model across the entire range of data for about a month after the storm. This is the version of the model that only matches chunks for that specific outlet and relies only on its history. Performance is very poor on the day following the storm because of the lack of relevant data, but improves very quickly, with even a single day worth of data, because of the importance of the recency factor. Performance actually regresses slightly after that, as outlets become available again in a pattern that is difficult to predict, especially without access to semantic data such as outlet brands, which might get resupplied at the same time.

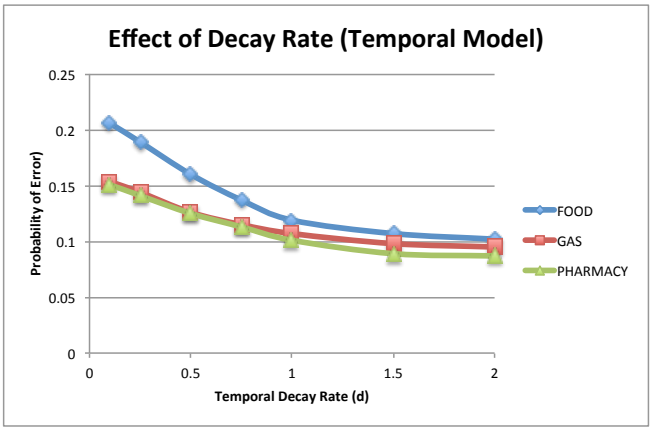


Figure 3: Effect of Decay Rate on Temporal Model.

Performance especially degrades on day 8, following a secondary storm that disrupts the pattern again. After that, it gradually improves over time to about 10% errors. Following the strong suggestion of the importance of the recency effect, Figure 3 examines the performance of the temporal rate as a function of the power law decay rate *d* for each outlet category averaged over all days, separated by outlet category. In general, a higher decay rate results in a lower error rate, indicating the primacy of recency over frequency. The availability of food outlets tends to be harder to predict than gas or pharmacy outlets, perhaps because their merchandise is more important or more perishable, leading to faster depletion, but the pattern is similar.

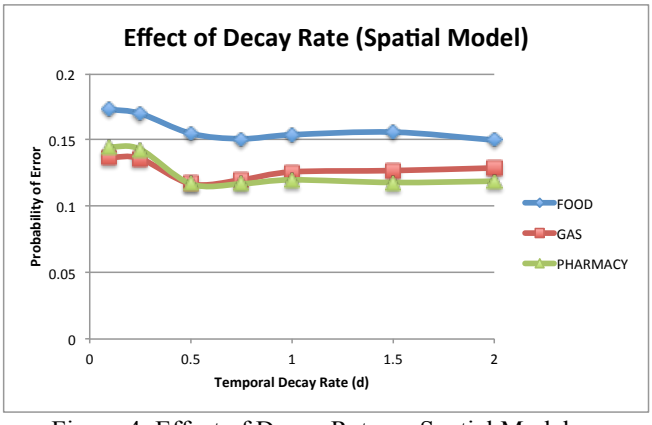


Figure 4: Effect of Decay Rate on Spatial Model.

Figure 4 reports the effect of the same decay rate, but for the spatial[3] model, that also reflects generalization across outlets using the partial matching and blending mechanisms. One can see that there is now a penalty for very high decay values that overemphasize recency. When considering a broader knowledge base, frequency of occurrence becomes more important and balances out against recency around the decay rate value of 0.5 that has become the standard value in ACT-R models for capturing human performance.

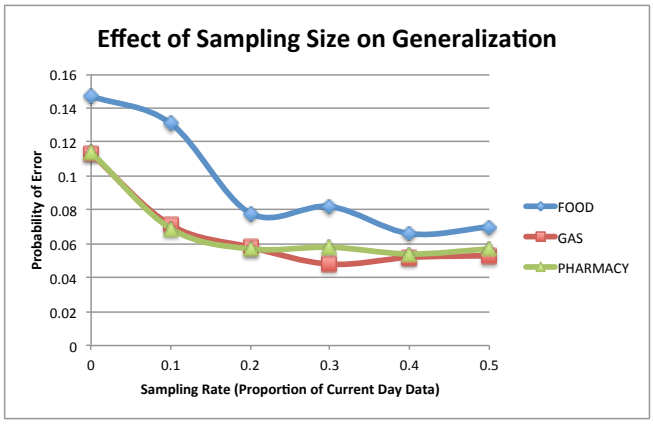


Figure 5: Effect of Sampling Size on Spatial Model.

The results of the spatial model presented in Figure 4 are actually slightly worse than those of the temporal model because we evaluated them on common ground, i.e., without including any of the current day's data for the spatial model to generalize from. Figure 5 examines the impact of the sampling rate of data for the current day to determine the effectiveness of the spatial model to generalize from nearby outlets. Generalization is quite effective, reducing the probability of error by half with about 20% of the current data. Note that more data (up to 50%) doesn't improve generalization further because of the overall unpredictability of the task, at least in certain conditions.

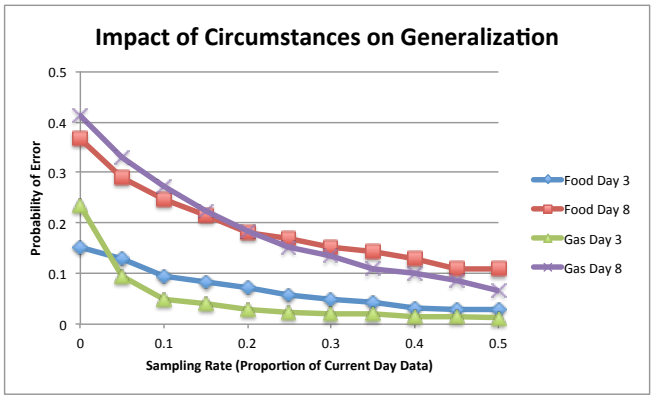


Figure 6: Effect of Circumstances on Spatial Model.

[3] We could refer to it as the integrated model because it also includes the temporal aspect through the base-level component, but we found the spatial/temporal distinction to be more descriptive.

Finally, to examine the impact of conditions on generalization, Figure 6 focuses on performance on Day 3 (2 days after the storm) and Day 8 (the day after a secondary storm hit) for food and gas outlets (pharmacy outlets omitted but results similar to gas). Because of the difficulty of predicting availability immediately after a disruptive event, the error rate is consistently and significantly higher on Day 8 than Day 3. However, as for the average across all days, performance significantly improves with sampling rate, becoming almost error-free on Day 3, which relies only on a single day of useful complete data (Day 2) and the specified proportion of the current day's data.

Discussion

Gu et al. (2014) applied a variety of algorithmic approaches to the prediction problem using this data set. They similarly differentiated their approaches between spatial and temporal algorithms. Their algorithms can be seen as specialized version of the cognitive mechanisms used here, e.g., the LastKnownState algorithm is simply the recency component of base-level activation without frequency, while the BestProxy algorithm is effectively partial matching without blending (or stochasticity). Recognizing the need to reflect both temporal and spatial data, they develop an algorithm that combines the best of the two approaches, in a way similar to, but more limited than, how those factors are combined in chunk activation.

The compelling argument for cognitive models, however, is not that they outperform a given machine learning algorithm. Rather, it is that they provide a way to augment human cognition in a way that is fundamentally compatible with it, for example by selecting a limited set of data to provide to the human decision maker that would result in the best human performance. In ongoing work, we are exploring mechanisms to introspect into the mechanisms of our cognitive model to drive data selection that would maximize its performance. We plan to then verify the model's predictions by collecting data in situations that combine model data selection and human decision making.

Acknowledgments

This research was supported by the Network Science Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory or the U.S. Government.

References

AI Magazine (2016). Beyond the Turing Test. AAAI Press. April, 2016.

Anderson, J. R. (1990). *The Adaptive Character of Thought*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y . (2004). An integrated theory of the mind. *Psychological Review* 111, (4). 1036-1060.

Cowan, N. (2001). "The magical number 4 in short-term memory: A reconsideration of mental storage capacity". *Behavioral and Brain Sciences* **24** (1): 87–114; discussion 114–85.

Erev, I., Ert, E., Roth, A. E., Haruvy, E., Herzog, S., Hau, R., Hertwig, R., Stewart, T., West, R., Lebiere, C. (2010). A choice prediction competition, for choices from experience and from description. *Journal of Behavioral Decision Making 23(1)*: 15-47.

Gu, S., Pan, C., Liu, H., Li, S., Hu, S., Su, L., Wang, S., Wang, D., Amin, T., Govindan, R., Aggarwal, C., Ganti, R., Srivatsa, M., Bar-Noy, A., Terlecky, P., & Abdelzaher, T. (2014). Exploitation Data Extrapolation in Social Sensing for Disaster Response. The *10th IEEE International Conference on Distributed Computing in Sensor Systems* (DCOSS 2014), Marina Del Rey, CA.

Kurzweil, R. (2006). *The Singularity is Near.* New york, NY: Viking Press.

Lebiere, C. (1999). The dynamics of cognitive arithmetic. *Kognitionswissenschaft* [Journal of the German Cognitive Science Society] Special issue on cognitive modelling and cognitive architectures, D. Wallach & H. A. Simon (eds.)., 8 (1), 5-19.

Lebiere, C., Gray, R., Salvucci, D. & West R. (2003) Choice and Learning under Uncertainty: A Case Study in Baseball Batting. In *Proceedings of the 25th Annual Meeting of the Cognitive Science Society*. pg 704-709. Mahwah, NJ: Erlbaum.

Lebiere, C., Pirolli, P., Thomson, R., Paik, J., Rutledge-Taylor, M., Staszewski, J., & Anderson, J. R. (2013). A Functional Model of Sensemaking in a Neurocognitive Architecture. *Computational Intelligence Neuroscience.*

Marr, D.; Poggio, T. (1976). "From Understanding Computation to Understanding Neural Circuitry". Artificial Intelligence Laboratory. A.I. Memo. Massachusetts Institute of Technology. AIM-357.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63 (2): 81–97.

Oaksford, M. & Chater, N. (2007). *Bayesian Rationality: The Probabilistic Approach to Human Reasoning.* Oxford University Press, Oxford, UK.

Stocco, A., Lebiere, C., & Samsonovich, A. V. (2010). The B-I-C-A of biologically inspired cognitive architectures. *International Journal of Machine Consciousness,* 2(2), 171-192.

Wallach, D., & Lebiere, C. (2000). Learning of event sequences: An architectural approach. In Proceedings of *International Conference on Cognitive Modeling* 2000, pp. 271-279. NL: Universal Press.

West, R. L., & Lebiere, C. (2001). Simple games as dynamic, coupled systems: Randomness and other emergent properties. *Journal of Cognitive Systems Research*, 1(4), 221-239.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Considerations Influencing Human TSP Solutions and Modeling Implications

Brandon S. Perelman (Brandon.S.Perelman.CTR@mail.mil)
U.S. Army Research Laboratory, Human Research & Engineering Directorate
Aberdeen Proving Ground

Shane T. Mueller (ShaneM@MTU.edu)
Michigan Technological University, Department of Cognitive and Learning Sciences,
1400 Townsend Drive Houghton, MI 49931

Abstract

The visual Euclidean Traveling Salesman Problem (TSP) presents participants with nodes, representing cities, and requires that the participant trace the shortest closed route among the cities. Humans solve a similar problem in every day navigation and search tasks. We investigated human TSP solutions for considerations other than solution length. We found a preference for solutions favoring distance-discounted reward and distance to first contact. A hierarchical stochastic model parameterizing solution length, distance-discounted reward, goodness of fit, and plan complexity showed similar effects. The model shows promise for approximating human performance in TSP and other TSP-like naturalistic tasks.

Keywords: TSP; planning; problem solving; visual cognition.

Introduction

The Traveling Salesman Problem (TSP) is a spatial combinatorial optimization problem used in various forms in applied settings, such as operations (e.g., the Vehicle Routing Problem; Dantzig & Ramser, 1959) and engineering (Krolak, Felts, & Marble, 1971), and basic research on spatial cognition and navigation in animals (de Jong, Gereke, Martin, & Fellous, 2011) and humans (Tenbrink & Seifert, 2011). Visual Euclidean TSP requires that the solver plot the shortest path through a 2D metric space containing nodes, representing cities, beginning and ending in the same location. TSP is computationally intractable, with each problem having $(n - 1)! / 2$ solutions. Therefore, brute-force approaches to obtaining optimal, shortest path solutions are too resource-intensive for many applications.

Despite the aforementioned complexity of TSP, human solutions to TSP are typically an order of magnitude shorter (i.e., better) than those produced by many heuristic algorithms (MacGregor & Ormerod, 1996), and are typically no more than 10% longer than the optimal solutions, increasing linearly with problem size (Dry, Lee, Vickers, & Hughes, 2006; MacGregor & Ormerod, 1996; Pizlo et al., 2006). Because human solutions are fast and near-optimal, understanding the mechanism people use to generate them has implications for algorithm development.

Evidence suggests that humans do not exhaustively solve the problem at initial presentation. For example, Kong and Schunn (2007) showed that participants perform the majority of their global information-seeking saccades after beginning to solve the problem. Mueller, Perelman, Tan, and Thanasuan (2015) found very short (~4s) planning times (interval between initial viewing and beginning to solve the problem) that increased linearly with problem size.

These characteristics have prompted the suggestion that humans use a hierarchical approach to problem solving in which a rapidly formed global plan guides the local decisions (e.g., Best & Simon, 2000). Many computational accounts of TSP follow this hierarchical structure, simplifying the problem space by grouping individual cities into clusters (e.g., Pizlo et al., 2006) or designating a global path through the space that starts as a convex hull (MacGregor, Ormerod, & Chronicle, 2000).

This same strategy of following a rapidly produced global plan is likely used in similar tasks. One such task, searching for a target among candidate locations, requires planning a route that optimizes a distance-discounted reward function to minimize the estimated time to find (ETF) that target (see Wiener, Schnee, & Mallot, 2004). This general task is critical in operational domains, such as wilderness search and rescue (Perelman & Mueller, 2013) and military and public safety search operations (Antoniades, Kim, & Sastry, 2003), as well as for sports such as orienteering (Blum et al., 2007). The present study investigates the extent to which a single adaptive mechanism could be used to solve TSP and other TSP-like problems.

Investigations of human behavior in naturalistic TSP-like tasks (e.g., Blum et al., 2007; Perelman & Mueller, 2013; Perelman & Mueller, 2015; Ragni & Wiener, 2012; Tenbrink & Seifert, 2011; Tenbrink & Wiener, 2009) suggest that real-world strategic planning requires considering factors other than path length. Many of these tasks require solvers to prioritize ETF and distance to first contact (DFC), optimizing a function that rewards visiting locations early in the path, versus TSP where rewards are uniform. Wiener et al. (2004) suggest that certain cluster-based strategies should produce this behavior, and Tenbrink and Wiener (2009) found a slight bias (roughly 58% of solutions; 8% more than expected) toward prioritizing larger over smaller clusters early in solutions to a naturalistic TSP-like task. Note that we have termed these alternative solution criteria 'considerations' rather than heuristics as they are not necessarily isolated mechanisms, but components of an underlying mechanism.

If a common mechanism is used to solve TSP and similar tasks, then we should see evidence of these alternative considerations in TSP solutions. To our knowledge, there is no published literature searching for evidence of these considerations in traditional TSP solutions. Evidence of a common mechanism holds implications for algorithm

development and our understanding of human visual problem solving.

General Method

The present study consists of analyses of three datasets - two derived from experiments presented here, and one generously donated by other authors (see below).

Experiments (Datasets) 1 and 2

Michigan Technological University students participated in Experiments 1 and 2 (n = 29 and 35, respectively). The goal of Experiment 2 was to replicate the results of Experiment 1 using a blocked design to reduce potential fatigue.

Two participants in Experiment 1 provided incomplete data (final n = 27). Participants completed TSP problems presented using the Psychology Experiment Building Language v. 0.14 (PEBL; Mueller, 2014) TSP. PEBL TSP problems begin in a fixed starting location, with the last segment automatically completed by the software, and route edits are not allowed (see Mueller et al., 2015).

Participants in Experiment 1 completed 5 6-city practice problems, then 15-problem sets presented in random order, each containing 10, 20, or 30 cities for 50 total trials. Participants in Experiment 2 completed their trials in 2 blocks, each containing 5 6-city practice problems, then 5-problem sets each containing 10, 20, and 30 cities presented in random order, for a total of 40 trials between both blocks.

Dataset 3

Data for this analysis were provided by Chronicle, MacGregor, Lee, Ormerod, and Hughes (2008). In that study, 110 University of Hawaii students completed 9 30-city problems by connecting the cities using pen and paper; the data were converted into electronic format manually.

Analyses

Traditional descriptive statistics of solution lengths relative to optimal are reported for the first two experiments. We used a novel method, reverse solution analysis, to investigate the solutions for bias toward ETF and DFC. ETF was operationalized here as the cumulative sum of all segments in the solution weighted by serial order of visitation. DFC is defined as the length of the first segment.

Reverse Solution Analysis

TSP solutions begin and end on the same city; they are closed loops. Therefore, a solution and its reverse form are equal in solution length (the only consideration by which solutions are evaluated in TSP). However, the two solutions may differ in terms of ETF or DFC (or other criteria of solution quality). We report bias toward a given consideration in the observed distribution when the percentage of solutions superior to their reverse forms with respect to that consideration exceeds that of the expected distribution in which both forms appear with equal frequency. The observed percentage over the expected percentage (50%) indicates the magnitude of the bias.

Because the PEBL TSP script automatically completes the last segment of solutions, these biases will be calculated for both the closed and open solutions, which omit the final segment returning to home. Note that Dataset 3 was generated using a paper and pencil format and required a return to home, and was included to show the extent to which automatic solution completion impacts performance.

Results and Discussion

Solution Length

Solution length provides a strong measure of overall efficiency (Figure 1). In Experiment 1, across all problem sizes, participants' mean solution lengths were 5.30% longer than optimal (*S.D.* = 8.73%). Between set sizes, efficiency degraded with increasing problem size. Solutions to the 20-city problems showed the highest variance in solution length, an effect which was mirrored in the Experiment 2 results indicating that this likely reflects something about the city configurations for those problems.

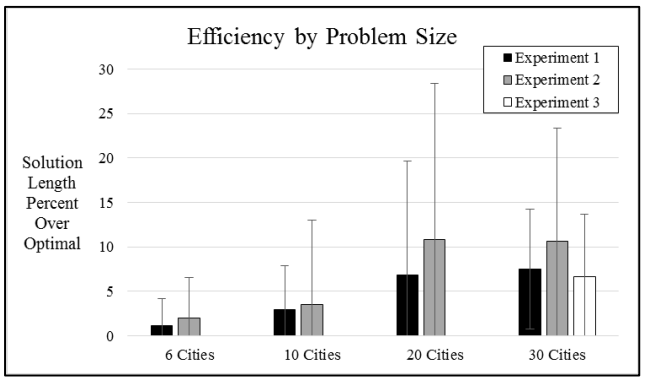


Figure 1: Efficiency by Problem Size for All Experiments. Error bars indicate standard deviation.

Experiment 2 efficiencies were largely consistent with those seen in Experiment 1. On average, participants' mean solution lengths were 6.76% longer than optimal (*S.D.* = 12.67%). Between problem sizes, solution lengths increased with problem size, though efficiency in the 20- and 30-city problems was not notably different. Finally, participants' efficiency in solving the 30-city (*M* = 6.67%, *S.D.* = 7%) problems in Experiment 3 was consistent with Experiment 1 performance for problems of the same size.

Other Solution Considerations

Of all Experiment 1 solutions (n = 1,358), 317 were optimal whereas 93 were designated poor (operationalized as 15% longer than optimal). Table 1 shows ETF and DFC bias by solution quality, measured using reverse solution analysis. Each block of cells indicates the percentage of solutions that favor that particular criterion given solutions of equal length.

The ETF bias block presents solutions quantified as either closed (complete, as generated by participants) or open (without the return to home). These results indicate the

presence of ETF bias in the complete solutions. The magnitude of this effect appears consistent with that observed by Tenbrink and Wiener toward prioritizing larger versus smaller clusters earlier in the solution (2009; 58%). However, this effect is smaller when considering the solutions without the return to home. No clear trend in ETF bias was observed with respect to solution quality. DFC bias was also detected in these solutions, and the effects are likely related.

Table 1: Experiment 1 RSA Results, ETF and DFC Biases

ETF Bias

	Solution Quality	Percent Favoring ETF	Binomial Test Significance
Closed Solutions	**Optimal Solutions**	64.98	$p < .001$*
	All Solutions	63.40	$p < .001$*
	Poor Solutions	70.97	$p < .001$*
Open Solutions	**Optimal Solutions**	56.80	$p = .018$*
	All Solutions	52.30	$p = .098$
	Poor Solutions	45.16	$p = .417$

DFC Bias

	Solution Quality	Percent Favoring DFC	Binomial Test Significance
Closed Solutions	**Optimal Solutions**	67.19	$p < .001$*
	All Solutions	68.56	$p < .001$*
	Poor Solutions	80.65	$p < .001$*

To visualize this effect, we plotted the proportional distance of the solution covered by each segment in serial order (Figure 2). ETF and DFC biases are evidenced by shorter moves earlier in the solutions, or on the first move, respectively. Figure 2 shows reasonably uniform segment lengths for all except the final segment, indicating that most (but not all) of the bias effect appears to be explainable by a failure to account for the return to home cost.

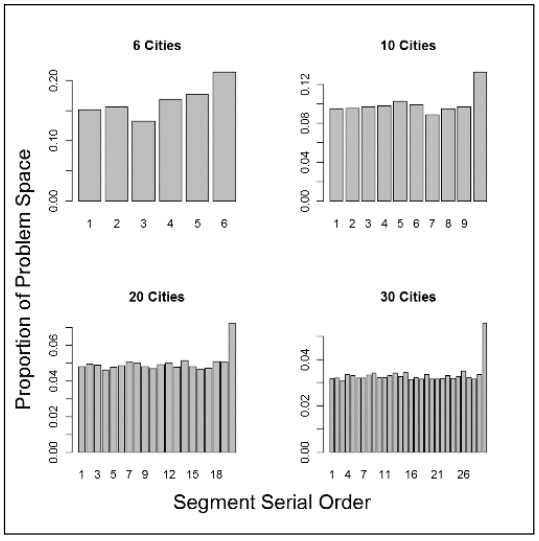


Figure 2: Proportion of problem space covered by each segment across Experiment 1 problems, by problem size.

The results of Experiment 1 show a robust bias toward solution forms that visit cities earlier in the solution at the expense of costs associated with the return to home.

In Experiment 2, participants only appeared to fail to account for the return to home on the larger 20- and 30-city problems, as indicated by their long final segment lengths (Figure 3). Aggregated across problem sizes, the results were similar to those seen in Experiment 1 with the exception that ETF bias disappeared entirely for the open solutions, and was not related to solution quality (Table 2).

Table 2: Experiment 2 RSA Results, ETF and DFC Biases

ETF Bias

	Solution Quality	Percent Favoring ETF	Binomial Test Significance
Closed Solutions	**Optimal Solutions**	59.84	$p < .001$*
	All Solutions	60.21	$p < .001$*
	Poor Solutions	60.74	$p = .008$*
Open Solutions	**Optimal Solutions**	51.71	$p = .539$
	All Solutions	51.21	$p = .378$
	Poor Solutions	46.01	$p = .347$

DFC Bias

	Solution Quality	Percent Favoring DFC	Binomial Test Significance
Closed Solutions	**Optimal Solutions**	55.64	$p = .031$*
	All Solutions	65.14	$p < .001$*
	Poor Solutions	75.46	$p < .001$*

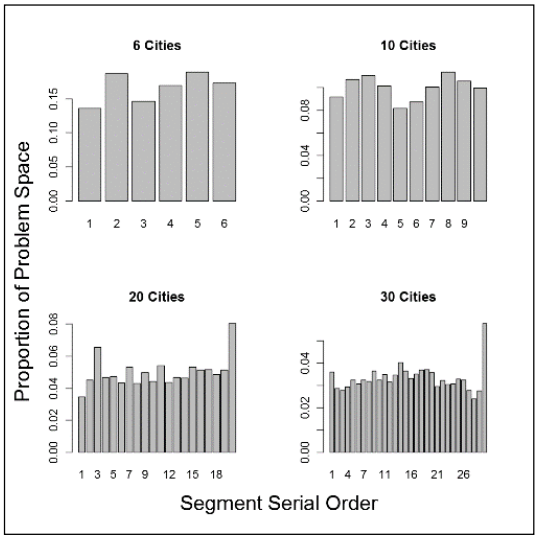


Figure 3: Proportion of problem space covered by each segment across Experiment 2 problems, by problem size.

Visualizing these data at a coarser grain size reveals ETF bias in the 6-city problems, and a trend toward it in the 10-city problems, with longer moves generally appearing in the second halves of the solution (Figure 4), despite no clear failure to return home.

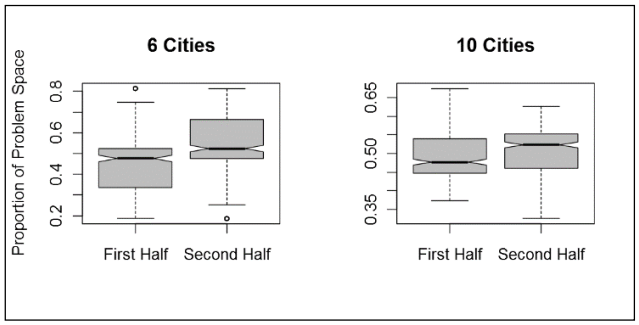


Figure 4: Distance covered by each half of the solution, Experiment 2. Lower values in first half indicate ETF bias.

One potential explanation for this effect is that it is an artifact of the experimental software. The PEBL TSP automatically completes solutions, so it is possible that participants' failure to account for the return home arises from the fact that they are not required to complete this section of the solution. Therefore, analysis of Dataset 3 tested for this effect in a paper and pencil version of TSP.

Dataset 3 consisted of 975 solutions to 9 30-city problems. 56.82% of these solutions favored the ETF-superior form (binomial test, $p < .001$) with 63.18% of solutions favoring the DFC-superior form (binomial test, $p < .001$). Figure 5 shows a strong failure to account for the return to home cost in all but one problem (Problem 3044).

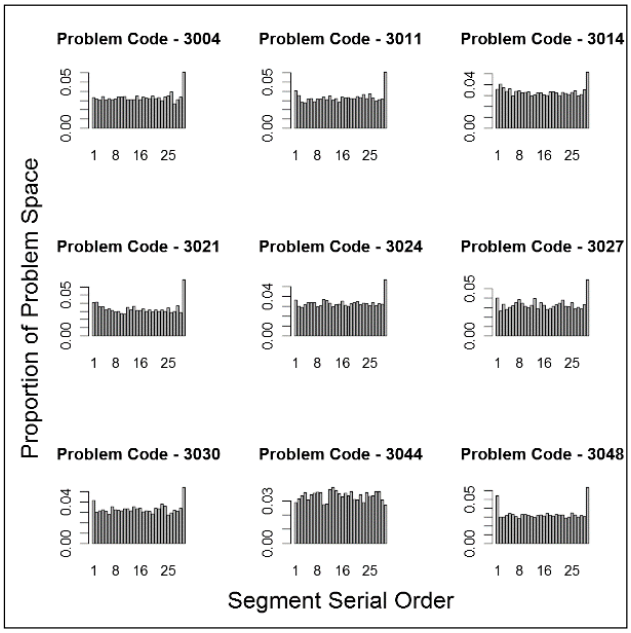


Figure 5. Proportion of problem space covered by each segment across each of the Dataset 3 problems.

To estimate the magnitude of the effect of the DFC bias and the failure to return home in each of these experiments, we divided the first and final segment lengths, respectively, by the average segment lengths (Table 3). One-way ANOVAs found significant effects of final / average segment length by solution quality for both Experiment 1, $F(1, 1356) = 94.85$, $p < .001$, and Experiment 2, $F(1, 953) = 165.1$, $p < .001$.

Table 3: First, Final / Average Relative Segment Length by Solution Quality, Mean Percent (*S.D.*)

	Solution Quality	Experiment 1	Experiment 2	Dataset 3
First Segment	Optimal Solutions	94.06 % (56.04 %)	85.86 % (52.82 %)	120.13 % (88.30 %)
	All Solutions	98.01 % (63.55 %)	88.45 % (62.53 %)	
	Poor Solutions	100.24 % (91.07 %)	101.89 % (96.91 %)	
Final Segment	Optimal Solutions	126.89 % (61.98 %)	98.03 % (60.89 %)	176.25 % (132.83 %)
	All Solutions	155.54 % (98.90 %)	148.41 % (111.09 %)	
	Poor Solutions	259.48 % (175.10 %)	226.37 % (171.83 %)	

Note: Solutions not aggregated by quality for Dataset 3 as optimal solutions to these problems were not available.

For Experiments 1 and 2, the average final segment length ranged from slightly shorter to over 2.5 times as long as the average segment length, with the paper and pencil TSP (Dataset 3) producing results falling somewhere in the middle. For Experiments 1 and 2, the final segment length generally increased as solution quality degraded, with the optimal solutions having much shorter final segment lengths relative to average than the poor solutions.

Similar effects were not observed for first / average segment lengths in Experiment 1, but the effect of solution quality on first / average segment lengths was observed in Experiment 2, $F(1, 953) = 6.89$, $p = .008$, with the better solutions producing shorter first segment lengths relative to average. First segment lengths in Dataset 3 were longer than average, though a causal mechanism is not readily apparent.

Finally, in Experiments 1 and 2 the larger problem sizes generally produced longer first and final segment lengths relative to average (see Table 4). One way ANOVAs revealed significant effects of problem size on first, $F(1, 1048) = 11.79$, $p < .001$, and final, $F(1, 1048) = 102.60$, $p < .001$, segment lengths in Experiment 2. The effect of problem size on segment length was observed for the final, $F(1, 1356) = 20.66$, $p < .001$, but not first, $F(1, 1356) = 0.07$, $p = .799$, segment lengths in Experiment 1.

Table 4: First, Final / Average Relative Segment Length by Problem Size, Mean Percent (*S.D.*)

	Problem Size	Experiment 1	Experiment 2
First Segment	6	96.97 % (71.42 %)	83.61 % (57.00 %)
	10	97.49 % (59.66 %)	93.44 % (49.89 %)
	20	98.26 % (68.03 %)	69.56 % (61.06 %)
	30	96.06 % (62.04 %)	110.68 % (81.85 %)
Final Segment	6	145.34 % (77.70 %)	113.43 % (77.11 %)
	10	143.07 % (78.29 %)	102.76 % (58.71 %)
	20	152.12 % (102.05 %)	175.07 % (141.70 %)
	30	175.39 % (129.40 %)	183.84 % (137.20 %)

Human Results Summary and Discussion

The results presented above demonstrate, for the first time to our knowledge, the presence of considerations pertinent to naturalistic TSP-like tasks in traditional TSP solutions. Participants (1) produced solutions that reduce distance to first contact and (2) preferred visiting locations early in the solution at the expense of overall solution length, resulting in higher distance-discounted reward. However, (3) this effect is driven largely by a preference for solution forms with a longer return to home. (4) This effect is robust to the test delivery format (i.e., computer with an automatic return to home versus manual pen and paper format) and (5) the magnitude of this bias is related to the quality of the closed solutions – better solutions reduce the discrepancy between final and average segment lengths.

Points 1 and 2 suggest that humans solve a more general problem than TSP task instructions would require; in addition to solution length, humans consider ETF and DFC during problem solving. And, in light of the constraint of point 3, they seem to do so at the expense of task performance, though better human solutions tend to consider the problem space more globally, therefore accounting for the return home.

Subject matter expert interviews (Perelman, 2015) indicate that ETF and DFC are critical for certain tasks, and prior research (Perelman & Mueller, 2015) has shown that humans can adapt their solution criteria to fit specific tasks. However, the results of the present study show that even when tasked with minimizing solution length in a traditional TSP, humans still generate solutions that account for considerations relevant in naturalistic spatial problem solving tasks. This suggests a common mechanism used for both TSP, and for naturalistic TSP-like tasks. A simple computational model was developed by Perelman (2015) to describe adaptive behavior in TSP-like problems, which we apply here to investigate an underlying adaptive mechanism capable of producing the above effects in TSP.

Modeling

In light of the human results, we used a computational model designed to permit this flexibility in strategic control that is capable of solving TSPs using limited at-a-glance information about the problem space. The goal of this model was to use a scheme capable of adapting to task requirements (i.e., it incorporates strategic considerations such as ETF into solution planning) to reproduce human efficiency dynamics and solution characteristics, specifically the bias toward ETF-superior solution forms.

The model uses a two-layer hierarchical structure: a computationally inexpensive local decision making algorithm (nearest neighbor) guided by a general *plan* (see Figure 6) that considers multiple criteria that can be tailored to specific goals and tasks (i.e., path length minimization, as in TSP, versus discounted-rewards used in naturalistic tasks). This higher level plan representation consists of a small number of segments; the model solves for all the cities within each segment in sequence. The plan is initially drawn by running K-means clustering over the problem space (k = 6) then connecting the cluster centroids from the starting position by nearest neighbor.

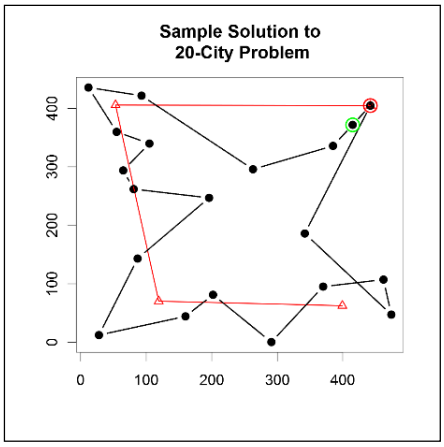


Figure 6. The higher level plan (red segments) guides local solutions (black lines) among the cities (black dots) from the start location (red; first move is green to show direction)

The plan is then iteratively fit to the data by minimizing a cost function comprised of a linear combination of five weighted parameters, (1) log number of segments, intended to represent plan complexity, (2) goodness of fit, the average distance between a plan segment and its constituent cities, (3) plan length, (4) distance-discounted reward, the sum of the path lengths of all segments discounted by their serial order, and (5) the average angle between segments, intended to penalize doubling back. This plan is fit to the data using 500 iterations during which a point in the plan is added, deleted, moved, or swapped in serial order with another. This optimization process was not intended to duplicate that used by humans to solve the problem, but rather to demonstrate that a model that incorporates multiple criteria can account for some patterns in the human data.

Modeling Results

The present model solved the 10-, 20-, and 30-city TSP problems used in Experiment 1 20 times each, and the solutions were analyzed using the methods described above.

Solution Length

To evaluate model solution efficiency, solution lengths were compared to those of the optimal solutions. Across all problem sizes, model solutions lengths were 16.08% longer than optimal (*S.D.* = 11.56%). Interestingly, as with the human subjects, the model produced the greatest variance in solving the 20-city problems (Figure 7), indicating that human performance on these problems can be attributed to properties of the problem spaces rather than a fluke in our particular sample. While the model was less efficient than humans, it displayed similar dynamics in the present problem set for the change in solution quality at the experimental problem sizes, and variance within set sizes.

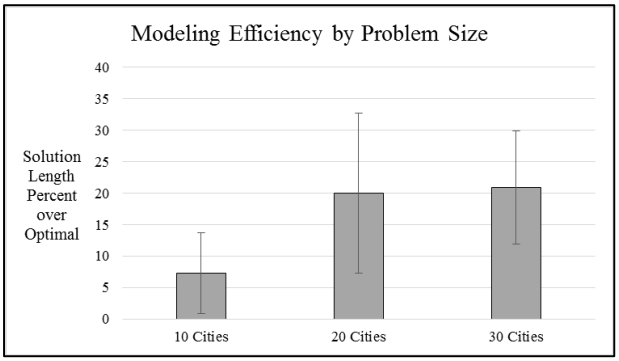


Figure 7. Model solution length by problem size. Error bars indicate standard deviation.

The second goal for our model was to replicate the human bias toward DFC- and ETF-superior solution forms. We investigated the model solutions using the method applied to the previous experiments and found that the model favored ETF- and DFC-superior solutions more strongly than humans (Table 5), preferring a biased solution in nearly every case except for the 20-city problems. Figure 8 demonstrates qualitatively the effects of ETF and DFC bias in the model's solutions, along with a failure to account for the return to home producing segments roughly twice as long as average.

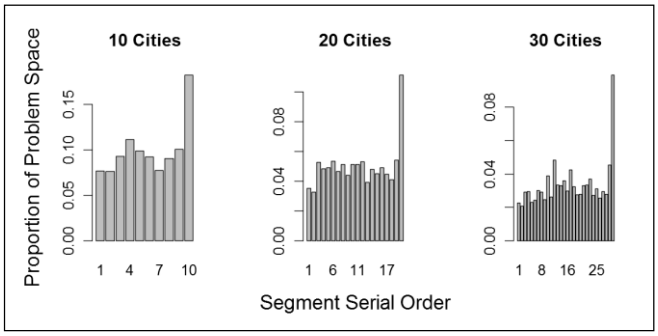


Figure 8: Proportion of problem space covered by each segment across Experiment 1 problems, by problem size.

As with the human solutions, we further quantified the ETF and DFC biases, and the effect of the failure to return home, by comparing the first and final segment lengths, respectively, to the lengths of the average segments on those trials. The model produced solutions with values (Table 5) similar to those of the poor human solutions (Table 3), including comparatively short first segment lengths, and long final segment lengths (i.e., the return to home) that increased with problem size. However, unlike the human solutions, the model produced solutions with first segment lengths that decreased with increasing problem size relative to the average segment lengths on those trials. Finally, the model produced solutions with shorter relative first segment lengths, but longer final relative segment lengths, compared to the human solutions.

Table 5: Final / Average Segment, Mean Percent (*S.D.*), and Proportion of Solutions Favoring Each Bias

	10 Cities	**20 Cities**	**30 Cities**
First / Average Segment Length	79.16 % (41.91 %)	70.85 % (31.93 %)	67.95 % (25.90%)
Final / Average Segment Length	205.85 % (56.19 %)	257.52 % (114.72 %)	327.16 % (138.09 %)
Percent Favoring ETF	99.33 %	89.67 %	100 %
Percent Favoring DFC	100 %	94.33 %	100 %

Last, we investigated a potential criticism of the present model – that a nearest neighbor model would be equally efficient. We compared model and human solution lengths to those generated using nearest neighbor. The present model produced solutions to the 10-city problems that were 4.1% shorter than nearest neighbor, equal in length for the 20-city problems, and 1.1% shorter for the 30-city problems. Human solutions in Experiment 1 generally showed a similar pattern (6%, 10% and 10%, respectively), though the poor human solutions were on average 3-9% longer than those produced by nearest neighbor. In summary, the model solutions were less efficient than human solutions on average, but more efficient than the nearest neighbor and poor human solutions.

General Discussion

The behavioral results of the present study show, for the first time, that human TSP solutions consider ETF and DFC, criteria that are irrelevant to TSP, but critical in the real world. This manifested here as a failure to account for the return to home, and the magnitude of this bias was related to solution quality – poor solutions had longer final segments.

A model that adjusts a linear plan to fit the problem space according to a number of criteria related to TSP, real-world TSP-like problems, and plan complexity, exhibits behavior similar to humans in this task. We expect that efficiency could be greatly improved via dynamic re-planning to match the human data. In adapting the present model, the agent would adjust its higher level plan after solving for the points within each segment in serial order. In this way, a parameter estimated from prior eyetracking studies (e.g., Kong & Schunn, 2007) would govern the iterations spent in dynamic replanning.

Taken together, the results of the present study hold implications for modeling human performance in spatial combinatorial optimization problems. Specifically, the results speak to the importance of granularity and sequence in representing the problem space. Many algorithms, such as those implementing a convex hull, solve the problem exhaustively at presentation. The behavioral and modeling results presented here are consistent with prior work suggesting that humans approach these problems using a mechanism that provides a means of solving the problem efficiently without the mental burden of generating and maintaining an exhaustive solution in memory, at the expense of efficiency later in the route. Finally, this mechanism is consistent with producing solutions to discounted-reward problems that are more common than path length optimization in naturalistic tasks.

References

Antoniades, A., Kim, H. J., & Sastry, S. (2003, December). Pursuit-evasion strategies for teams of multiple agents with incomplete information. In *Proceedings of the 42nd IEEE Conference on Decision and Control, 1,* 756-761.

Blum, A., Chawla, S., Karger, D. R., Lane, T., Meyerson, A., & Minkoff, M. (2007). Approximation algorithms for orienteering and discounted-reward TSP. *SIAM Journal on Computing*, *37*, 653-670.

Chronicle, E. P., MacGregor, J. N., Lee, M., Ormerod, T. C., & Hughes, P. (2008). Individual differences in Optimization Problem Solving: Reconciling Conflicting Results. *The Journal of Problem Solving*, *2*(1), 41-49.

De Jong, L. W., Gereke, B., Martin, G. M., & Fellous, J. (2011). The traveling salesrat: Insights into the dynamics of efficient spatial navigation in the rodent. *Journal of Neural Engineering, 8.* doi:10.1088/1741-2560/8/6/065010.

Dry, M., Lee, M. D., Vickers, D., & Hughes, P. (2006). Human performance on visually presented traveling salesperson problems with varying numbers of nodes. *Journal of Problem Solving, 1,* 20-32.

Dantzig, G. B. & Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6,* 80-91.

Kong, X. & Schunn, C. D. (2007). Information seeking in complex problem solving. In *Proceedings of the 8th International Conference on Cognitive Modeling*, Oxford: UK, 261-266.

Krolak, P., Felts, W., & Marble, G. (1971). A man-machine approach toward solving the Traveling Salesman Problem. *Communications of the ACM, 14,* 327-334.

MacGregor, J. N. & Ormerod, T. C. (1996). Human performance on the Traveling Salesman Problem. *Perception & Psychophysics, 58,* 527-539.

MacGregor, J. N., Ormerod, T. C., & Chronicle, E. (2000). A model of human performance on the traveling salesperson problem. *Memory & Cognition, 7,* 1183-1190.

Mueller, S. T. (2014). PEBL: The Psychology experiment building language (Version 0.14) [Computer experiment programming language]. Retrieved June 2014 from http://pebl.sourceforge.net.

Mueller, S. T., Perelman, B. S., Tan, Y., & Thanasuan, K. (2015). Development of the PEBL Traveling Salesman Problem Computerized Testbed. *The Journal of Problem Solving*, *8*, 4.

Mueller, S. T. & Piper, B. J. (2014). The Psychology Experiment Building Language (PEBL) and the PEBL Test Battery. *Journal of Neuroscience Methods, 222,* 250-259. Doi: 10.1016/j.jneumeth.2014.10.024.Perelman, B. S. & Mueller, S. T. (2013). A Neurocomputational approach to modeling human performance in simulated unmanned aerial search. In *Proceedings of the 12th International Conference on Cognitive Modeling*, Ottawa, CA.

Perelman, B. S. (2015). A naturalistic computational model of human behavior in navigation and search tasks. *Doctoral Dissertation*.

Perelman, B. S. & Mueller, S. T. (2015). Identifying mental models of search in a simulated flight task using a pathmapping approach. In *Proceedings of the 18th International Symposium on Aviation Psychology*, OH.

Pizlo, Z., Stefanov, E., Saalweachter, J., Li, Z., Haxhimusa, Y., & Kropatsch, W. G. (2006). Traveling salesman problem: A foveating pyramid model. *The Journal of Problem Solving*, *1*, 8.

Ragni, M. & Wiener, J. M. (2012). Constraints, Inferences, and the Shortest Path: Which paths do we prefer? In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Sapporo, Japan.

Tenbrink, T. & Seifert, I. (2011). Conceptual layers and strategies in tour planning. *Cognitive Processes, 12,* 109-125.

Tenbrink, T. & Wiener, J. (2009). The verbalization of multiple strategies in a variant of the traveling salesman problem. *Cognitive Processes, 10,* 143-161.

Wiener, J. M., Schnee, A., & Mallot, H. A. (2004). Use and interaction of navigation strategies in regionalized environments. *Journal of Environmental Psychology*, *24*, 475-493.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

The Representation of Visual Working Memory

Bella Z. Veksler, Rachel Boyd
(bellav717@gmail.com) (rachel.boyd.rb1@gmail.com)
Oak Ridge Institute for Science & Education at AFRL

Christopher W. Myers, Glenn Gunzelmann
(christopher.myers.29@us.af.mil) (glenn.gunzelmann@us.af.mil)
Air Force Research Laboratory, Wright-Patterson AFB, OH, USA

Hansjörg Neth (h.neth@uni-konstanz.de)
University of Konstanz, Konstanz, Germany

Wayne D. Gray (grayw@rpi.edu)
Rensselaer Polytechnic Institute, Troy, NY, USA

Abstract

Visual working memory (VWM) is a construct hypothesized to store a small amount of accurate perceptual information that can be brought to bear on a task. Much research concerns the construct's capacity and the precision of the information stored. Two prominent theories of VWM representations have emerged: slot-based and continuous-resource mechanisms. Prior modeling work suggests that a continuous resource that varies over trials with variable capacity and a potential to make localization errors best accounts for the empirical data. Questions remain regarding the variability in VWM capacity and precision. Using a novel eye-tracking paradigm, we demonstrate that VWM facilitates search and exhibits effects of fixation frequency and recency, particularly for prior targets. Whereas slot-based memory models cannot account for the human data, a novel continuous-resource model provides a better fit and identifies the relevant resource as item activation.

Keywords: visual working memory; visual search; ACT-R.

Introduction

Visual working memory (VWM) is a construct hypothesized to be a limited capacity system that maintains representations of visual information for temporary storage and manipulation for ongoing tasks (Luck & Vogel, 2013). This construct has garnered much attention and has been the focus of many studies and computational models. Even so, answers to fundamental questions, such as its capacity and representation precision, remain elusive (van den Berg & Ma, 2014).

Two theories of VWM representations dominate the literature: *slot* and *continuous resource* mechanisms. Slot theories generally posit a fixed capacity of 3 to 4 items with high to perfect precision (Luck & Vogel, 2013). A slot is a discrete memory container filled with an object representation with bound visual features (Luria & Vogel, 2011). Information stored within a slot can be accurately applied to a task regardless of its visual complexity, be it a single vertical line or a complex Chinese character.

By contrast, continuous resource theories of VWM posit a finite resource that can be spread across different areas of a scene or item. This resource is seen as a pool of mental processing power dedicated to VWM, which can be flexibly distributed across items in a display (Wilken & Ma, 2004). Fewer objects to be encoded lead to less distributed memory resources and allow for more precise object representations.

Recently, Donkin, Kary, Tahir, and Taylor (2016) have argued that a VWM system using a continuous resource may appear to support a slot interpretation when the number of items to remember varies from trial to trial. At times highly precise representations of a small number of objects appear to favor a slot-based model, but when set size is unpredictable participants are biased to focus on a small subset of items, leading to performance suggestive of a slot model. When set size was predictable (the same across multiple trials), resource models best characterized the data.

Van den Berg, Awh, and Ma (2014) varied precision, capacity, and the potential for spatial binding errors as three independent factors of VWM to test lingering questions. Using a 4x4x2 factorial design, all models were tested on 10 previously published empirical results from a change detection paradigm. The results indicated that a continuous model which varied both storage capacity and precision across trials, combined with the presence of the potential for spatial binding errors best accounts for the data. However, questions regarding the mechanisms behind the variance in precision and capacity remain unanswered.

A passive, tachistoscopic version of the change detection paradigm has been the dominant approach to establishing prominent theories of VWM (Alvarez & Cavanagh, 2004), and continues to be the paradigm most used in contemporary empirical research on VWM (Donkin et al., 2016). In this task, a participant is instructed to attend to and remember information within a stimulus display. The information is typically a set of unique objects that differ across features, such as shape and color. After some time the stimulus disappears and after a delay the object of the possible change is cued, or a new stimulus appears. If a change occurred, the participant must indicate the change in some manner, either by responding yes/no (c.f., Alvarez & Cavanagh, 2004), by identifying what has changed (c.f., D. E. Anderson, Vogel, & Awh, 2013), where the change occurred (c.f., Barton, Ester, & Awh, 2009), or some combination thereof. Researchers

vary the number of items in a stimulus (i.e., set size) to evaluate VWM capacity and use change identification to evaluate VWM precision.

There are weaknesses in the passive change detection approach to understanding VWM (Rouder, Morey, Morey, & Cowan, 2011). Specifically, many, if not all, VWM studies rely on a passive approach to understanding VWM rather than an active one (Findlay & Gilchrist, 2003). Outside the experimental laboratory, visual search does not occur in a vacuum, but rather in the context of a task where targets contained in some visual array are distinguished from distractors. We argue that passive change detection with delayed responses (~2–3 s) does not tap into the functional importance of VWM — to facilitate the accurate completion of an active visual task through the temporary storage of readily available and accurate visual information.

In the current paper we provide an explanation for the variance in VWM precision and capacity. To do so, we introduce a new eye-tracking paradigm that moves away from the change detection tasks commonly used to investigate VWM. Our new paradigm of *repeated serial search* (Neth, Gray, & Myers, 2006) requires an individual to actively search for different (and sometimes repeating) targets within a stable visual display and thus represents a task that is more realistic and ecologically valid than a passive change detection paradigm. Importantly, it allows us to ask questions of VWM that inform how VWM drives search behavior and the potential differences in depth of encoding between targets and distractors since we have access to the full history of fixations.

Our empirical and modeling work leads to five important conclusions: (1) the variability in VWM capacity results from recency and frequency effects from selectively encoding visual information; (2) VWM precision variability results from the same recency and frequency effects; (3) memory facilitates search behavior; (4) targets have a stronger mnemonic trace than distractors; and (5) the relevant "resource" involved is memory activation. In the following sections we introduce our paradigm and present empirical results, followed by a model analysis of the empirical data.

Experiment

To determine the degree to which VWM facilitates visual search, we designed an experiment using a novel *repeated serial search* paradigm. In this paradigm, participants were required to search the same spatial configuration of 10 static items a total of 20 times. This paradigm taps into the VWM construct, motivating participants to retain a maximum amount of information in VWM to facilitate future searches.

Paradigm. On each trial, ten circular objects with a diameter of 60 pixels were distributed randomly over a centered white rectangular display area (measuring 1270-by-970 pixels). The objects were positioned at least 60 pixels away from any edge and the distance between the centers of any two objects was constrained to be at least 200 pixels. Each circle contained a hidden label (upper case letter, number, or monosyllabic four-letter word) that specified the target sought by the participant. On any given trial only one type of label was in the circles (letters, numbers, or words). The order of label types was randomized within each participant's task presentation.

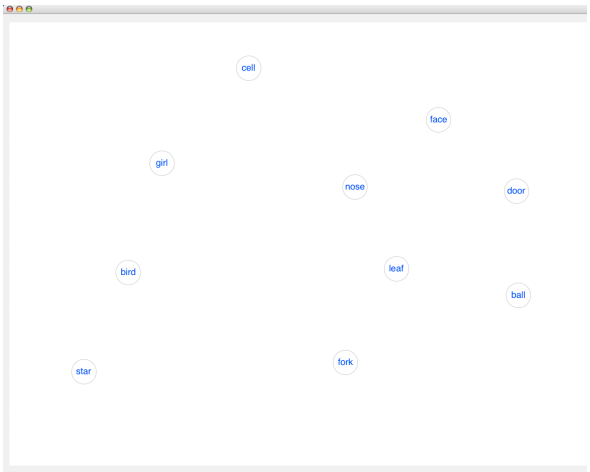


Figure 1. Example stimulus used in the experiment. Although all labels are visible here, they were hidden from participants' view until a cursor hovered within the circle.

Each trial was composed of 20 searches through the display. At the beginning of each search, the experimental software announced the current target label to the participant (e.g., "cell" in Figure 1). Participants could hover with the mouse cursor over each circle to uncover its hidden label. Once the cursor was moved off the circle, the corresponding label was hidden again. Participants were instructed to click on the circle corresponding to the target label. If the clicked circle indeed contained the correct target label, a new target was announced; however, if a clicked circle contained a different label the software recorded an error and the current target was announced again to provide a reminder to the searcher. Consequently, searchers typically uncover non-targets (distractors) in the process of searching for targets and these distractors may turn into targets in subsequent searches.

There were three within-participants information presentation types that manipulated the number of intervening targets between identical targets. While they are of theoretical interest, we collapse across these presentation types for the current analyses to save space and mitigate complexity.

Participants. A total of 13 Rensselaer Polytechnic Institute undergraduates (3 females) volunteered for course credit. Their mean age was 18.92 years ($SD = 1.04$).

Procedure. Participants signed informed consent forms, viewed a slideshow of the instructions, and were calibrated to an LC Technologies eye tracker prior to beginning the study. Every participant completed 60 trials in total. Each trial consisted of a series of 20 searches. Every search commenced when a computer generated voice announced the next target to be found.

Results

A timeline of the sequence of fixations during every search within a trial was created for each participant. This was made possible through the collection of visual point of regard and mouse click data while participants performed the visual search task. Given this sequence of fixations, we can determine the frequency of fixations across labels and how long ago — in terms of duration and the number of intermediate items — each label was last viewed to investigate recency and frequency effects in finding a target. We can also determine differences that are due to the functional role of labels (i.e., whether labels were previously seen and encoded as targets or as distractors).

Recency effects. For this analysis we restricted the data to the first two times an item was a target of a search. A 2 (label-type)-by-10 (recency) ANOVA was performed to evaluate the effect of label encoding and recency of last fixation. There was an interaction between whether an item was a target before and how recently it was last fixated, $F(9, 108) = 3.76, p < .001$, η^2=0.24. There was also a significant main effect of recency, $F(9, 108) = 11.94, p < .001$, η^2=0.50 (see Figure 2 top). This effect was greater for labels that had not been previous targets, $F(1, 12) = 73.42, p < .001$, η^2=0.86. In general, labels that were prior targets were less impacted by recency of fixation.

One explanation for the inverted U-shape of the items that were only distractors prior to the current search is that as participants are searching the display, they are only encoding whether or not the current item is the target, rather than the identity of the item. This depth of encoding may result in an inhibition of return effect for more recently fixated items (2–5 fixations ago) leading to longer search times than when the distractor was seen longer ago.

Frequency effects. A 2 (label-type)-by-7 (frequency) AVOVA was performed to evaluate the effect of label encoding frequency. There was insufficient data in frequency bins 1 and 2 (i.e., in cases where a second search for a target was preceded by one or no fixations on the item prior to the search), leaving bins 2–8 for analysis (see Figure 2 bottom). Nonetheless, these bins reflect the general trend in the data. There was a significant interaction between fixation frequency and label-type on the number of fixations to find the target, $F(6, 72) = 5.92, p < .001$, η^2=0.33, where searches required fewer fixations when a label had been a target before despite being seen less than 5 times, $F(1, 12) = 38.37, p < .001$, η^2=0.76, (Figure 2 bottom). Further, there was a main effect of frequency on number of fixations to find the target, $F(6, 72) = 3.25, p < .01$, η^2=0.21. In particular, items that were not prior targets show a benefit of having seen the item more times, whereas items that were previously targets seem to be encoded sufficiently enough that it takes roughly the same number of fixations to find the target regardless of the number of previous fixations.

Recency and frequency effects. In order to provide a more robust description of the human data, we examined the

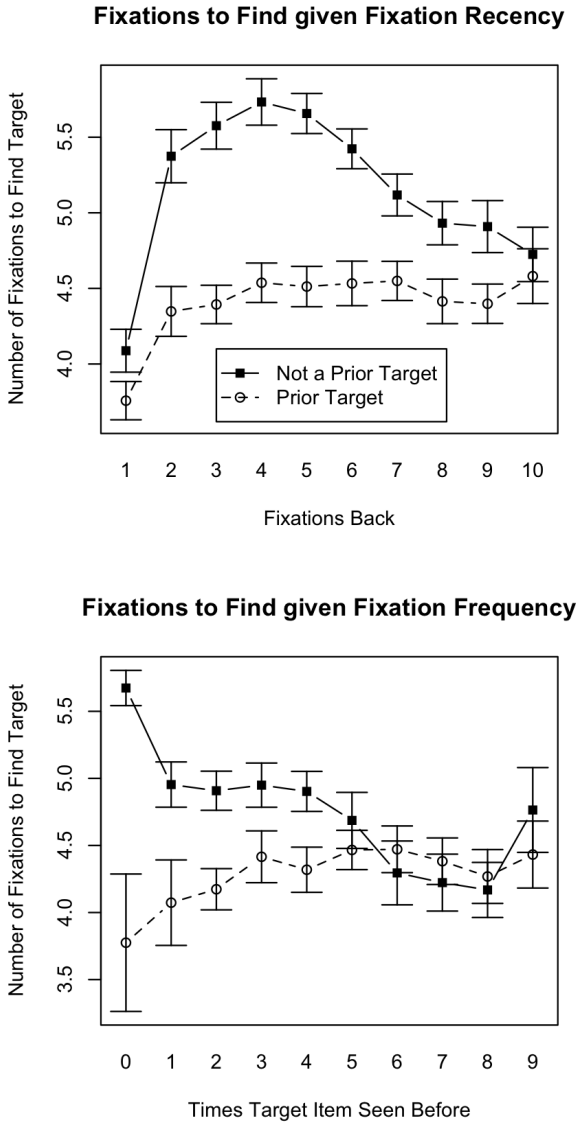


Figure 2. Mean number of fixations needed to find a target as a function of recency and frequency of seeing the target before.

proportion of all searches in which a target was last seen *R* (Recency) fixations ago or was seen *F* (Frequency) times prior to the search and was found within *N* fixations. Figure 3 illustrates the respective distributions generated by analyzing the human data in this way. In particular, in the recency graph, the peak of each distribution shifts to the right (more fixations to find target) as *R* increases. It should be noted that the human data exhibits a bell-shaped curve across all recency values, with the proportion of searches in which target is found in a higher number of fixations falling off gradually.

In the frequency graph, when the item has never before been fixated (*F*==0), the proportion of all searches in which the target is found stays at roughly 10% across all *N*. Items which have been seen more times (*F*==9) have a slightly

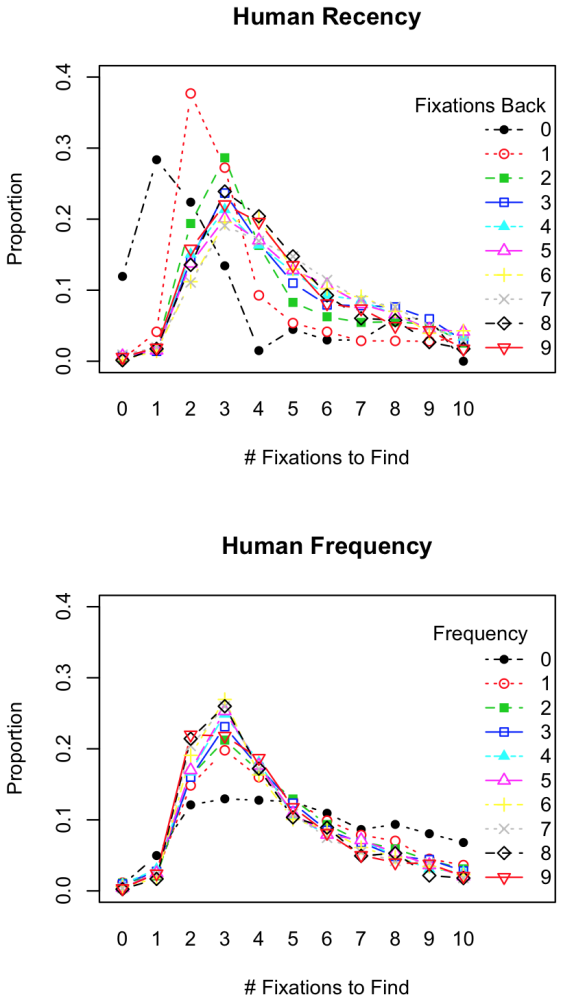


Figure 3. Proportions of recency and frequency effects in the human data.

more pronounced peak at N=3 as compared to items which were fixated fewer times.

Subsequent model runs were compared on the basis of these distributions. We wanted to be able to capture both the magnitude of the proportions in both recency and frequency, as well as the general shape of the distributions as proportions gradually tapered off for the higher N. Note that this collapses across whether or not the item was a prior target.

Experiment Discussion

The results from the study indicated that the number of fixations to find a target is affected by (1) whether that label had been a prior search target, (2) the recency of a label's previous fixation, and (3) the frequency of a label's previous fixations. Each of these effects contributes to the variability in VWM capacity and precision. A label more recently encoded will lead to the appearance of a larger VWM capacity and higher VWM precision. Similarly, a label more frequently encoded will lead to the appearance of a a larger capacity with greater precision. In passive change detection, the probe is chosen at random and may sometimes select a target that has neither been recently or frequently encoded. This could naturally lead to the perception of capacity and precision variability of VWM. By looking at the selective attention process during the search, we can more concretely point to the mechanisms leading to this variability.

Model-based Analysis

Given the debate in the literature between slot based and continuous resource models of VWM, we chose to run a factorial combination of models and search strategies. The three classes of models were: No memory, Slot-Based Memory, and Continuous Resource Memory. The search strategies were either Nearest First or Random. For each memory-strategy combination, scan-paths were generated for each of the 20 searches within a trial. In all models, the assumption is that once an object was visited, it was removed from the set of possible next visits until the next target was announced.

No Memory Model

This model served as a theoretical baseline for the other models and searched the display for every search within a trial without any memory for previous targets or distractors. In the *random search* version, the model searched the display in a random fashion. In the *nearest first* version, the model allocated attention to the closest object to the one currently being fixated. No parameters were varied in this model.

Slot-Based Memory Models

This class of models had a slot-based memory and the number of slots available ranged from 0 to 10. Slots were instantiated as a queue (FIFO) based on the human fixation history prior to the current search (see Figure 4). Uncovering a label re-instantiates slot 0 and pushes the labels contained in slot i into slot $i + 1$. This corresponds closely to the R denotation in the Recency human data analysis. At the beginning of every search, the model queried its slot-based memory to determine whether the target was already present in one of the slots. If it was, the model immediately directed its attention to the location of the target. If it was not in one of the slots, the model searched the display in either a Random or Nearest First manner. Only the number of slots parameter was varied in this model type.

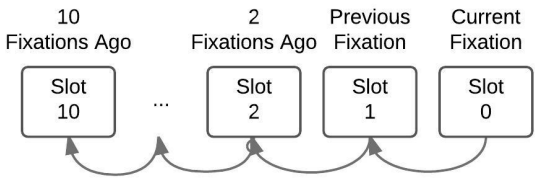


Figure 4. Slots are instantiated corresponding to the timeline of fixations in the human data.

Continuous Resource Memory Models

This class of models relied on human eye fixation history of the trial prior to the current search, taking into consideration both the time stamps of when the item was fixated and how many previous fixations were made to the item. The ACT-R memory equation (J. R. Anderson, 2007) was applied to each of the items. It specifies the activation of a given item i in memory as

$$A_i = ln \sum_{j=1}^{n} t_{ij}^{-d} + \beta_i + \epsilon_i \quad (1)$$

where j is a fixation on the item and t_{ij} is a time stamp of how long ago the item was seen on fixation j, $-d$ is a decay value, β is a base level constant offset, and ϵ is logistically distributed transient noise with a mean of 0 and standard deviation of σ.

Activation of the target item was recalculated at the beginning of each search and the model checked whether the activation of the target was above threshold, T, and if so, moved attention directly to the known location of the item. If $A_{target} < T$, then the model selected and encoded another item based on either a *random search* or a *nearest first* strategy. If the target item was still not found, activation was recalculated for the target at each additional movement of attention.

Four parameters were varied in the context of ACT-R's memory equation (d, β, T, and σ) to find the best fit to the human recency and frequency data using MindModeling.org (Harris, 2008). In particular, we varied the parameters as follows: d: [0,1], β: [0,10], T: [0,20], and σ: [0,5]. This created a total of 27,951 combinations of parameters for each search strategy.

Model Evaluation

Each of the above models was run through all trials (and searches) obtained from human data. The ACT-R memory equation uses recency and frequency information as sources of activation for a given chunk in memory. Thus, we examined the human data as a function of both the recency and the frequency of previous fixations to current targets. In this case, recency refers to how many fixations ago the item was last fixated, R with respect to the current fixation. For each parameter set, summary statistics were calculated to determine the percentage of all trials on which the target was seen R fixations ago or was previously seen F times and found in N fixations. This resulted in 10 distributions for recency and another 10 for frequency, each with 11 data points (one for each N of fixations to find the target, see Figure 3 for human data). Then Root Mean Squared Error (RMSE) and R^2 scores were calculated for each target recency curve and for each target frequency curve.

A composite goodness-of-fit measure was created to combine the R^2 and RMSE measures to capture both the shape and the magnitude of the differences between human data and model predictions. Because best fits according to R^2 are values closer to 1, and best fits according to RMSE are values closer to 0, we re-scaled the R^2 measure (1-R^2) and computed an average of all curves for each parameter setting.

Table 1
Best fits for all model types.

Memory	Strategy	Composite Score*
Continuous resource	Nearest first	0.07
Continuous resource	Random	0.09
Slot (2)	Random	0.35
Slot (2)	Nearest first	0.36
None	Nearest first	0.37
None	Random	0.37

Note: *Lower composite scores indicate better model fits.

The best fitting slot-based model was one which contained 2 slots ('remembered' the last two items previously fixated; see Table 1). The best fitting continuous resource model resulted from the following parameter settings: d=1, β=1, T=10, and σ=4.0 (see Figure 5).

The no memory model established a baseline with which the other memory models could be compared. As can be seen in Table 1 and Figure 5, the continuous resource memory model did a much better job of capturing human performance. In particular, whereas a slot-based memory with 2 slots was the best fitting in this particular class of models, it failed to capture the shape of both the recency and frequency distributions. The continuous resource model, on the other hand, exhibited the bell-shape curve with gradual drop-off seen in the human data for both recency and frequency. Furthermore, a *nearest-first* search strategy was marginally better at capturing the effects than a *random search* model, suggestive of the type of strategy participants may have used as they conducted their search of the display.

We further evaluated the flexibility of all the model types to determine how convincing the fits actually are (and whether they could have been achieved merely by searching such a large space). Model Flexibility Analysis (MFA) was used to calculate the proportion of all empirical outcomes that each model could have potentially fit (Veksler, Myers, & Gluck, 2015). Although the *slot* model only has one parameter (number of slots), it's actually more flexible than the *continuous resource* model which has 4 parameters. MFA revealed flexibility for the *slot* model to be ϕ = .14 and for the *continuous resource* model to be ϕ = .014. Thus the *continuous resource* model makes more precise predictions and is less flexible.

Discussion & Conclusions

In the current work, we explored why variability in VWM capacity may at times exhibit variable precision and capacity. The new paradigm of *repeated serial search* allowed us to more readily observe the specific shifts of visual attention that occur during natural search. Human data suggests that the variability in VWM precision and capacity is closely tied to selective attention as search progresses.

Selective attention directly affects the ease with which subsequent targets can be found, with both recency and fre-

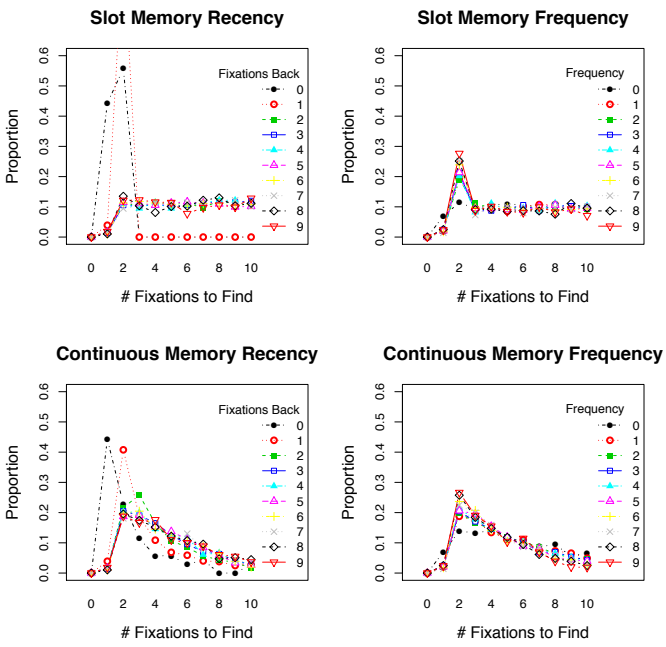


Figure 5. Model fits for best fitting slot and continuous resource models, nearest first search strategy.

quency playing a role. Items that were previously fixated more recently resulted in faster search times and boosted the likelihood of recalling the location of the target. Likewise, items which had previously been fixated more often were easier to find. Importantly, there was a stronger mnemonic trace for items which were previous targets as these items were found faster than those which were only fixated as distractors during previous searches.

We compared two models of VWM: a *slot*-based and a *continuous resource*-based model. In the case of the slot-based model, the recency of an item's encoding is taken into consideration to facilitate subsequent searches. However, this was not sufficient to account for the human data as it failed to capture the shapes of the distributions in both recency and frequency domains. A continuous resource model, on the other hand, directly incorporated both effects of selective attention. The continuous resource was instantiated as the item's activation, computed by taking into account both the frequency and recency of previous item fixations.

One limitation of the current approach is that none of the models explicitly account for the stronger mnemonic trace for prior targets. The continuous resource model could potentially account for this difference by including an item's fixation duration in its computation of activation - target items typically have longer fixations and more opportunity for rehearsal. While such models are beyond the scope of the current work, they are an interesting avenue for future research. Another possible concern is that humans may use an 'adaptive avoidance' strategy in which items known to *not* be the target are actively not gazed at. Future work will need to address the degree to which this type of strategy may drive behavior in visual search.

In conclusion, the repeated serial search paradigm elucidates the variability seen in VWM capacity and precision by taking into account selective attention considerations. Future work could apply the same continuous resource model to other data sets to explore the robustness of the model in accounting for various VWM results, as well as incorporating potentially hybrid models which combine slots and continuous resources.

References

Alvarez, G. A., & Cavanagh, P. (2004). The capacity of visual short-term memory is set both by visual information load and by number of objects. *Psychological Science*, *15*(2), 106–11.

Anderson, D. E., Vogel, E. K., & Awh, E. (2013). Selection and storage of perceptual groups is constrained by a discrete resource in working memory. *Journal of Experimental Psychology: Human Perception and Performance*, *39*(3), 824–835.

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?* (F. E. Ritter, Ed.). Oxford University Press.

Barton, B., Ester, E. F., & Awh, E. (2009). Discrete resource allocation in visual working memory. *Journal of Experimental Psychology: Human Perception and Performance*, *35*(5), 1359–67.

Donkin, C., Kary, A., Tahir, F., & Taylor, R. (2016). Resources masquerading as slots: Flexible allocation of visual working memory. *Cognitive Psychology*, *85*, 30–42.

Findlay, J. M., & Gilchrist, I. D. (2003). *Active vision: The psychology of looking and seeing*. Oxford Univ. Press.

Harris, J. (2008). Mindmodeling@home: a large-scale computational cognitive modeling infrastructure. In *Proceedings of the sixth annual conference on systems engineering research 2008* (pp. 246–252).

Luck, S. J., & Vogel, E. K. (2013). Visual working memory capacity: from psychophysics and neurobiology to individual differences. *Trends in Cognitive Sciences*, *17*(8), 391–400.

Luria, R., & Vogel, E. K. (2011). Shape and color conjunction stimuli are represented as bound objects in visual working memory. *Neuropsychologia*, *49*(6), 1632–9.

Neth, H., Gray, W. D., & Myers, C. W. (2006). Memory models of visual search - searching in-the-head vs. in-the-world. *Journal of Vision*, *5*(8), 8–9.

Rouder, J. N., Morey, R. D., Morey, C. C., & Cowan, N. (2011). How to measure working memory capacity in the change detection paradigm. *Psychonomic Bulletin & Review*, *18*(2), 324–30.

van den Berg, R., Awh, E., & Ma, W. J. (2014). Factorial comparison of working memory models. *Psychological Review*, *121*(1), 124–149.

van den Berg, R., & Ma, W. J. (2014). "plateau"-related summary statistics are uninformative for comparing working memory models. *Attention, Perception, & Psychophysics*.

Veksler, V. D., Myers, C. W., & Gluck, K. A. (2015). Model flexibility analysis. *Psychological Review*, *122*(4), 755–769.

Wilken, P., & Ma, W. J. (2004). A detection theory account of change detection. *Journal of Vision*, *4*(12), 1120–35.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

An Account of Interference in Associative Memory: Learning the Fan Effect

Robert Thomson (robert.thomson@usma.edu)[1, 2]
[1]United States Military Academy, West Point, NY 10928 USA

Anthony M. Harrison[2] (anthony.harrison@nrl.navy.mil)
J. Gregory Trafton[2] (greg.trafton@nrl.navy.mil)
Laura M. Hiatt[2] (laura.hiatt@nrl.navy.mil)
[2]Naval Research Laboratory, Washington, DC 20375 USA

Abstract

Associative learning is an essential feature of human cognition, accounting for the influence of priming and interference effects on memory recall. Here, we extend our account of associative learning that learns asymmetric item-to-item associations over time via experience (Thomson, Pyke, Trafton, & Hiatt, 2015) by including link maturation to balance associations between longer-term stability while still accounting for short-term variability. This account, combined with an existing account of activation strengthening and decay, predicts both human response times and error rates for the fan effect (Anderson & Reder, 1999). This represents the highest fidelity replication of a human experiment for modeling that we are aware of.

Keywords: associative learning; interference; cognitive models; fan effects

Introduction

Associative learning is an essential component of human cognition, thought to be part of many mental phenomena such as classical conditioning (Rescorla & Wagner, 1972), similarity judgments (Hiatt & Trafton, 2013), and memory recall (Thomson, Pyke, Trafton, & Hiatt, 2015). Despite its ubiquity, it is difficult to model directly due to its entangled ties to other aspects of cognition (e.g., memory decay).

Perhaps associative learning's most studied effect is that of *priming* (and its converse *interference*). Priming occurs when the retrieval of one memory facilitates the retrieval of another. Conversely, interference occurs when a memory primes multiple other memories instead of just the ones that are useful or relevant to the current situation. Those other memories are said to *interfere* with the useful one. When there is high interference, recognition accuracies are relatively lower and recognition response times relatively longer when compared to situations where there is low interference, ostensibly due to having lower overall activation in memory. Assuming that the degree of interference is positively correlated with the number of competing associations, then having more competing associations (i.e., a higher *fan*) will lead to relatively higher error rates and latencies than memories having relatively fewer competing associations. This effect is most popularly known as the *fan effect* (Anderson, 1974).

In this paper we will extend our account of associative memory embodied in a cognitive architecture (Thomson, Bennati & Lebiere, 2014) to account for the fan effect experiment. This account of associative memory has already successfully predicted the complicated results of a multi-trial free and serial recall task, including asymmetric contiguity effects that strengthen over time (Thomson et al., 2015). Here, we extend our theory to include link maturation to balance associations between longer-term stability while still accounting for shorter-term variability. We then use the theory as part of a cognitive model that performs the fan effect experiment using the same stimuli and presentation times as the human participants.

By doing this, we become the first theory of associative memory to explain how associations are learned and updated throughout the fan effect experiment. Previous models considered only associations at the end of the experiment (Anderson & Reder, 1999; Schneider & Anderson, 2012; Anderson, 1974; Rutledge-Taylor and West, 2008); our model enhances their understanding of associative memory by describing the process of how these end-state associations are reached.

Associative Learning in Memory Recall

Our account of associative learning is situated in the cognitive architecture ACT-R/E (Adaptive Character of Thought-Rational / Embodied; Trafton et al., 2013), an embodied version of the cognitive architecture ACT-R (Anderson et al., 2004). ACT-R is an integrated theory of human cognition in which a "production system operates on a declarative memory" (Anderson et al., 1998). In ACT-R, recall and latency depend on three main components: activation strengthening, activation noise, and associative activation. These three values are summed together to represent an item's total activation. When a recall is requested, the item with the highest total activation is retrieved, subject to a retrieval threshold; if no item's activation is above the threshold, the retrieval is said to *fail* and no item is recalled. The latency of the recall is also inversely correlated to the recalled item's activation.

Activation Strengthening

ACT-R's well-established theory of activation strengthening (also called *base-level* activation) has been shown to be a very good predictor of human declarative memory (Anderson et al., 1998; Anderson, 2007). Intuitively, activation strengthening depends on how frequently and recently a memory has been relevant in the past, and is calculated as:

$$B_i = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right) \quad (1)$$

where n is the number of times an element i has been accessed in the past, t_j is the time that has passed since the jth access, and d is the learning parameter, specifying an element's rate of decay. Importantly, this equation predicts that items that have occurred recently, or have been rehearsed more, are more likely to be recalled than those that have not.

Associative Activation

In our account, associative strengths are learned, strengthened, and weakened over time as new elements are learned or prior elements re-experienced. These associations are learned between relevant working memory items within temporal proximity to one another, leading from earlier to later items (Thomson, Bennati, & Lebiere, 2014). The strength of the learned association (or how strongly an existing association is increased) is influenced by the amount of time that passes between when the items were each in working memory. If one item is immediately followed by another in working memory, they will become very strongly associated; on the other hand, if an item has been out of working memory for a while before another is added, they will be only weakly associated. Additionally, associations are *asymmetric*; an association can be stronger from an item *i* to an item *j*, for example, than the association from item *j* to item *i* (or, there could be no association from item *j* to item *i* at all).

To balance the rate of associative learning between long-term stability and short-term variability, link maturation was included as an additional parameter. Associative link maturation slows the rate of strengthening and weakening based on the number of times the link has been used. This supports long-term stability of well-experienced associative links while allowing for rapid short-term learning of new associative elements. In neural networks, maturation is equivalent to the process of settling to reach a stable equilibrium (Wills et al., 2005; Eliasmith, 2005). Maturation is set using a logistic function:

$$M = 1 - \frac{1}{1+e^{-(\ln(timesInContext)*MaturationRate)}} \qquad (2)$$

The maturation rate controls the steepness of the curve, or, in other words, controls how quickly links will stabilize.

To compute associative strength from an item *j* to an item *i*, the learning mechanism computes an increment I_{ji}:

$$I_{ji} = lr * w * M \qquad (4)$$

where *lr* is a learning rate parameter, *w* is the weight of the increment determined by the strength of the items in working memory (scales from 0 to 1), *M* is maturation, and *R* is refraction. This increment is used to update link strength as follows:

$$S_{ji} = S_{jiPrior} * \left(1 - I_{ji}\right) + (I_{ji} * mas) \qquad (5)$$

where is the strength of the link from *j* to *i*, $S_{jiPrior}$ is the prior strength of S_{ji}, I_{ji} is the learning increment from above, and *mas* is a parameter controlling the maximum possible associative strength.

When a new link is learned or existing link updated that shares a source *j* with other existing links, then each of those other links are discounted proportionally to the *weight* that the original link is updated (e.g., S_{ji} is updated so S_{jk} is discounted):

$$S_{jk} = S_{jkPrior} * \left(1 - I_{jk}\right) \qquad (6)$$

where I_{jk} is computed using the *weight* from the link from *j* to *i,* but using the maturation *M* from the link from *j* to *k*. Equation 6 normalizes the amount link *j* to *k* is discounted based on the degree to which it has settled. This allows for newer links to rapidly change while providing for long-term stability for more mature links.

This discounting function attenuates link strengths consistent with interference accounts of memory. As more concepts compete in memory, the amount of associative strength from each concept is reduced. In a balanced environment, this discounting will approximate the statistical likelihood $P(i/j)$, which is the odds of perceiving or retrieving *i* immediately prior to *j*.

Armed with an understanding of our modeling framework, we now turn to the fan effect experiment itself.

The Fan Effect Experiment

To understand the fan effect, we consider Anderson and Reder (1999) classical fan experiment They capture the fan effect in a recognition task where participants begin by learning 48 pairs of people and places. Persons and places could appear in multiple pairs, and each pair was shown for five seconds. Then, during testing, participants respond yes (*target*) or no (*foil*) to whether presented statements were previously studied: the *person* is in the *place* (e.g., 'the *hippie* is in the *park*'). In the testing phase, participants were provided a monetary reward based on their total score. The score was computed by providing 1 point for each correct response, plus an additional point for each 100 ms of response times faster than 1500 ms. This induced a speed-accuracy trade-off into the experiment.

The experiment proceeded according to three phases: a study phase, drop-out training, and then a testing phase. In the study phase, each stimulus pair was presented once on the screen for 5 seconds. In the drop-out training phase, participants were presented with questions 'Who is in the *location*?' and 'Where is the *person*?' Participants had to respond with all persons associated with the location (or vice versa). Participants had to correctly answer all these questions for person and location to complete the phase. Participants completed two of these drop-out training phases. Finally, in the testing phase, participants would respond *yes* or *no* to queries 'the *person* was in the *location*' with participants receiving feedback on their response.

The experiment manipulated the test stimuli in two different ways. The first was to manipulate the fan of the persons and places. In this experiment, fan is the number of persons associated with a place, and vice versa. Fan is controlled by varying the number of persons in each place, or the number of places with each person (e.g., 'the ***hippie*** is in the *bank*' or 'the *soldier* is in the ***park***'). Here, the fan of one term (person/place) was fixed at 2, while the fan of the other term (place/person respectively) was varied to be either 2 (low-fan) or 4 (high-fan).

The second manipulation was to control the composition of the set of test stimuli shown to participants by manipulating different target and foil conditions. There were four target conditions: facilitation, interference, suppression and control. In the facilitation condition, each target (e.g., 'the *biker* is the *tower*') was 'facilitated' by being repeated 5 times each in the stimuli set. In the interference condition each target was repeated only one time in the stimuli set, and was considered

interference because the target's person or place overlapped with a target from the facilitation condition (i.e., 'the *biker* is in the *factory*', or 'the *doctor* is in the *tower*').

The other two conditions were the suppression and control conditions. In these conditions, each target appeared once in the stimuli set, and consisted of facts that were seen in the interference (but not facilitation) condition, such as *factory* and *doctor* in the above examples. Examples of suppression targets included: 'the *writer* is in the *factory*', and 'the *doctor* is in the *bank*'. Due to a particularity in the original study, there is limited difference between suppression and control stimuli, because the controls were designed such that they would functionally suppress stimuli from the suppression condition (e.g., 'the *monk* is in the *bank*'). They are different insofar as the suppression stimuli were effectively two steps removed from the facilitation condition, while the control trials were effectively three steps removed.

Foils were classified according to three conditions: high-frequency foils, which used person/place concepts from the facilitation condition but with novel pairings, and were repeated 4 times each in the stimuli set; low-frequency foils, which had novel pairings of person/place concepts from the interference, suppression, or control conditions and appeared once each in the stimuli set; and mixed foils, which created novel pairings using one high-frequency concept from the facilitation condition and one low-frequency concept from the interference, suppression, or control conditions and were repeated only once in the stimuli set. In total, there were 48 target sentences and 54 foil sentences in the stimuli set.

The test stimuli set was presented three times in successive blocks, and all stimuli were presented in each block. Feedback was provided for 1 second after participants' responses, with an additional 1 second inter-trial interval[1].

The results of this study were consistent with interference effects: there were longer latencies and more errors in the high-fan (i.e., fan of 4) conditions relative to the low-fan (i.e., fan of 2) conditions for both targets and foils, with both high-frequency (i.e., facilitation) targets and foils having relatively higher accuracy and quicker latencies than their corresponding low-frequency counterparts. They also predicted lower relative accuracy in the interference condition relative to the suppression and control conditions, and no difference between suppression and control.

Prior Modeling of the Fan Effect

There have been several attempts to mathematically model fan effects (Anderson & Reder, 1999). Most prominent is Anderson and Reder's (1999) model whose equations were grounded in the ACT-R cognitive architecture (Anderson and Lebiere, 1998). This model can be broken down into three related equations.

$$S_{ji} = S + \ln(1/fan_j) \quad (7)$$

$$A_i = B_i + \sum_j W_j S_{ji} \quad (8)$$

$$T = I + Fe^{-A_i} \quad (9)$$

Equation 7 describes the spread of activation (S_{ji}) from element j to i as a function of associative strength intercept S attenuated by the fan of j, which is the number of concepts to which j is associated. In Equation 7, $1/fan_j$ is a simplification of $P(i/j)$ assuming equal frequencies of i and j. In Anderson and Reder, frequencies were not equal, and were instead set ahead of time according to the objective probabilities in the model. Equation 8 relates an activation function A_i to the base-level activation from Equation 1, and the sum of spreading activation from Equation 7 multiplied by an attentional weigh W_j. Prior efforts set B_i to 0 on the assumption that the drop-out testing would balance out base-level activation between stimuli.

Finally, Equation 9 computes retrieval time T based on an intercept I, time scale offset F, and the activation function A_i from Equation 3. The estimates for each parameter were as follows: I was 1197 ms, F was 773 ms, S was 2.5 ms, and W was .33 (reflecting an even weighting of '*person*' '*in*' '*place*'). Using these parameters, Anderson & Reder report a strong correlation with response times, $r = .956$. This model did not attempt to fit error patterns.

This model, while successful, does not focus on modeling both latency and error rates, which we believe is an important part of understanding priming and interference in associative learning. While describing nicely the final average performance of participants, they provide little intuition for *how* participants learn the associations via experiencing the task. For instance, Equation 7 uses a fixed value for each condition that does vary.

In contrast, our approach grounds our account of associative learning within the larger ACT-R/E architecture along with the constraints it places on cognition (Trafton et al., 2013) by using a production system simulating the time-course of perception, encoding, retrieval, and response. Once base-level activation is included as a factor, then the time-course of stimulus presentation and training becomes important in determining overall accuracy and response time. This added fidelity (and complexity) may test assumptions made in prior modeling efforts, and also may provide new insights or hypotheses about how participants learn the task. To that end, our model performs the experiment analogously to participants, and learns associations over time. This supports our theory of associative memory explaining how associations are learned and adapt over time.

Learning the Fan Effect

The model starts with only background knowledge of the words used in the experiment and is equipped with the procedural knowledge necessary to perform the experiment. It has no underlying knowledge of the concepts of person and place, and thus has no knowledge of targets or foils. As we have said, the model is presented with the same experimental paradigm as the human participants.

The model uses the same procedural knowledge at the start of each phase to perceive the person and place concepts:

[1] This 1 s ITI was not listed in the Anderson & Reder (1999) paper, however it was reported in subsequent research.

when it sees two concepts on the screen together, then they are linked into a common concept-pair, with each constituent person and place priming the concept-pair. This concept-pair is represented in memory as a single chunk of information containing three features: *person*, *place*, and *was-target*. Was-target is a binary *true/false* decision, where *true* indicates that the stimulus was a *target* and *false* indicates that the stimulus was a *foil*.

The external environment is a simulated computer screen. When perceiving stimuli, the model randomly encodes one symbol first and then the other. The model categorizes these symbols according to two features: *person-symbols* and *place-symbols*. These representations are functionally identical and are distinguished only for lexical purposes to simply categorize incoming words. Since stimuli are of the form 'the *person* is in the *place*' we present person-symbols on the left of the display and place-symbols on the right of the display.

When a concept-pair is learned or updated, then the associative strengths between the person/place concepts and the pair is strengthened while the strengths between the concepts and their other related concept-pairs are weakened. Since concepts are related to more concept-pairs on high-fan trials than on low-fan trials, high-fan concepts tend to have lower associative strengths to their related concept-pairs than low-fan concepts. This lower associative strength predicts that concept-pairs involving high-fan concepts will have slower response latencies and increased error rates. Also, since high-frequency stimuli have been seen more often, their strengths will be stronger than low-frequency stimuli, however maturation controls the degree to which these stimuli increase in strength (e.g., a link seen 4 times as often is not 4 times as strong).

In the study phase, the model automatically encodes all concept-pairs as targets. After encoding the stimuli and generating a concept-pair representation, the model then repeats this encoding until the stimuli are no longer presented on the display, averaging 2-3 rehearsals over the 5 second presentation time.

In the drop-out training phase, the model perceives either a *person* or *place* and attempts to retrieve all *places* where that *person* is (or all *persons* in that *place*). If the model correctly perceives all required elements then the model moves onto the next stimulus, otherwise it studies those stimuli again and returns to the drop-out training. Once the model has successfully retrieved all elements in both run-throughs of the query phase, the test phase begins.

The test phase is the critical phase where all response times and error rates were recorded. Similar to the study phase, the model begins to encode the concept-pair for *person* and *place* as an analogue to perceiving: was *person* in the *place*? The model then attempts to retrieve any decision containing said *person* and/or *place*. This decision is a prior concept-pair stored in memory including person, place, and the critical *was-target* decision. A response is then generated according to the following criteria: 1) if the model is unable to retrieve any concept-pair due to all pairs being below threshold, then it responds *foil*; 2) if the model correctly retrieves the matching *person*, *place*, and *was-target* decision then it responds with the respective decision: *target* for *true* and *foil* for *false*; or 3) if the model retrieves a mismatching concept-pair containing one *person* or *place* but not both, then it assumes that the response is a *target*. After the model responds it receives feedback, which it uses to encode the correct concept-pair. This includes encoding foils, which allows the model to be capable of correctly retrieving that an item was a foil seen in an earlier testing phase. This is a unique behavior of our model and reflects the fact that participants cannot 'ignore' decisions they've made and must encode feedback that they've seen.

Finally, if the model was incorrect or had retrieved a mismatched element, then for the feedback period it rehearses the correct response. This process is repeated across all trials through three test phases.

In the testing phase, response times are recorded from the stimulus onset time until the model has responded with the appropriate decision (target or foil). It is important to note that as a fully-implemented production system model, the complete time to respond include two relatively fixed durations: approximately 600 ms to encode the stimuli from the display, and approximately 350 ms to prime the motor command and press the response key. This is in a similar range to the structural offset I of 1197 ms that Anderson and Reder (1999) used in Equation 9. This means that fan effects in latencies occur mainly in the approximately 200 ms – 800 ms timeframe where the concept-pairs are retrieved. It is the retrieval of the concept-pair that determines fan effects in both latencies and accuracy.

One final difference between the present model and prior efforts is that our model incorporates base-level activation B_i (see Equation 8), but replaces the S_{ji} from Equation 7 with our learned S_{ji} from Equation 5. As previously mentioned, base-level activation reflects the recency and frequency of use of elements, and it is not a given that base-level would be equivalent for items across the different target and foil conditions, especially for the high-frequency vs. low-frequency elements where frequency is necessarily varied.

Results

The present model was run for 200 iterations with the parameters described in Table 1 below. As is apparent from Figures 1 and 2, the model qualitatively captured fan effects in both accuracy and latency, respectively, reflected by slower response times and higher error rates for high-fan concepts compared to low-fan concepts. We also predicted relatively higher accuracy for high-frequency conditions (facilitation for targets and high-frequency for foils) to the rest of the conditions; lower accuracy in the interference condition relative to the control and suppression conditions, and no difference between suppression and control target conditions. One difference is that we predict a smaller average fan (.03 s instead of .09 s) than did Anderson and Reder (1999).

Table 1. Parameters used in Fan Effect Experiment

PARAMETER	VALUE
Base-Level Learning (B_i)	.4
Learning Rate (*lr*)	1.1
Maturation Rate (*M*)	.5
Maximum Associative Strength (*mas*)	7.25
Mismatch Penalty	4

The model fits target accuracy with an $r = .83$. Interestingly, the source of errors is different between targets and foils. Target errors are due to failures in retrieving any concept-pairs, whereas foil errors are due to confusing foils with previously seen similar stimuli.

Table 2. List of Average Base-Level Activation (B_i) and Average Associative Strength per Link (S_i) per Condition across Drop-Out Training and Testing.

	Drop		*Test 1*		*Test 2*		*Test 3*	
	B_i	S_i	B_i	S_i	B_i	S_i	B_i	S_i
F2	.44	1.33	.54	.97	.73	.91	.86	.82
F4	.91	.69	.87	.55	.83	.48	.97	.44
I2	.40	1.34	.25	.99	.59	.94	.59	.86
I4	.70	.70	.11	.58	.40	.60	.60	.47
S2	.32	1.35	.11	1.03	.67	1.00	.34	.95
S4	.41	.72	.26	.61	.53	.60	.40	.49
C2	.47	1.40	.08	1.17	.25	1.05	.35	.98
C4	.44	.71	.37	.59	.60	.52	.60	.52
L2	N/A	N/A	.13	1.06	.57	1.00	.40	.92
L4	N/A	N/A	-.65	.57	.29	.60	.47	.48
M2	N/A	N/A	-.57	1.08	.42	1.01	.44	.92
M4	N/A	N/A	-.15	.61	.38	.52	.25	.51
H2	N/A	N/A	.63	3.21	.75	3.22	.78	3.20
H4	N/A	N/A	.47	2.65	.55	2.69	.55	2.71

Discussion

The present model describes the emergence of fan effects in both accuracy and latency using a theory of associative memory including an account of interference by discounting link strengths. As more stimuli (persons or places) are presented together (or within a short temporal window) they interfere with each other's associative strength, reducing overall activation. This has the effect of lowering overall accuracy and increasing response times. While prior explanations (Anderson & Reder, 1999; Schneider & Anderson, 2012) have presented good fits to static human performance, the present model learns both base-level activation (reflected recency and frequency of use) and associative weights throughout the entire experiment (including the testing phase) and predicts the presence of fan effects in both latency and accuracy across all target and foil conditions.

An advantage of modeling the fan effect experiment at this higher level of fidelity is that we are able to assess some if the assumptions made in prior modeling efforts. Most interesting is that, while the assumption that base-level would be similar between high-fan and low-fan stimuli, while this was valid (see Table 2) in aggregate, base-levels were highly variable between conditions and throughout the task. The present model was able to qualitatively match to human performance with associative activation strong enough to compensate for the differences.

While not obvious when examining end-state models, the average activation of concept-pairs between conditions (see Table 2) changes throughout the testing phase. Many models assume a fairly plastic study/learning phase and a fixed testing phase; however, our model learns throughout the experiment. For instance, the added interference from learning novel foils reduces the activation of targets, especially in the interference condition.

A potential concern that our model addresses that was foils were not encoded in Anderson and Reder (1999) when they were perceived. It seems odd that stimuli seen in training were encoded while stimuli seen in testing were not. For instance, when a novel foil is perceived it does not increment the fan of targets. For instance, if a studied fan-4 target 'the *biker* is in the *factory*' was tested after perceiving the novel foil 'the *hippie* is in the *factory*' then that fan-4 place term '*factory*' should in fact be incremented to be a fan-5 place term. In our model, the notion of fan-4 or fan-5 is solely for classification purposes of the various conditions. Link

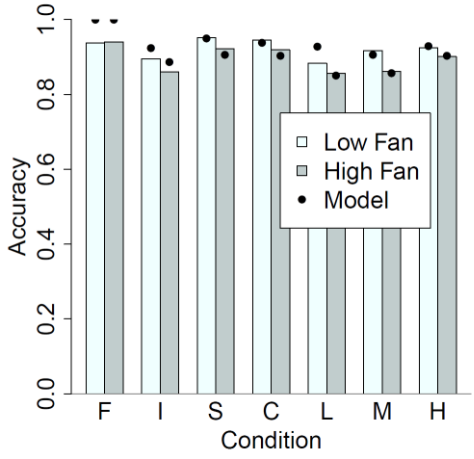


Figure 2. Error rates across target and foil conditions. Target conditions are Facilitation, Interference, Suppression, Control. Foil conditions are Low, Mixed, and High Foils.

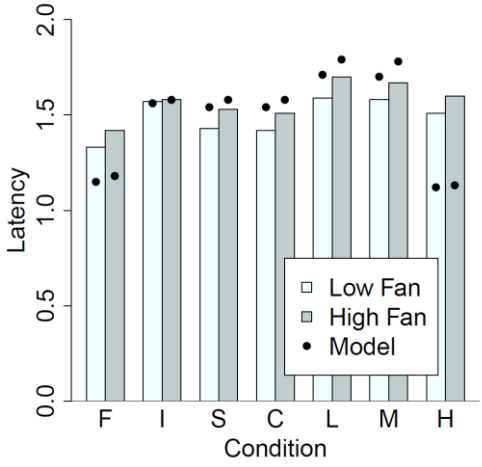


Figure 1. Latencies across target and foil conditions. Target conditions are Facilitation, Interference, Suppression, Control. Foil conditions are Low, Mixed, and High Foils.

strengths vary based on their use in the experiment. What is important is not whether an item is a fan-2 or fan-4 stimulus, but to what degree *biker* and *factory* prime the sentence (c.f., concept-pair) 'the *biker* in the *factory*.'

While the present model was able to predict fan effects in all conditions, it predicted a much faster response time for high-frequency targets and foils than humans exhibited. This was because the added frequency increased base-level activation too quickly. As seen in Table 2, associative strength was comparable between the high-frequency facilitation condition and the low-frequency interference/ suppression/control conditions. The difference was the higher base-level activation. The traditional activation equation (Equation 8) sums both base-level activation and associative strength, but it may be the case that the relative weighting of these factors changes over time based on some features of stimuli (such as relative strength, familiarity, or some other metacognitive feature). While the existing equation is well-justified in the literature, the inclusion of an adaptive frequency-based associative learning component replacing the fixed S_{ji} (Equation 7) may change the underlying balance between base-level and associative strength.

Another difference between the current model and human performance is that our model did not model speed-accuracy trade-offs reflecting the time-pressure based reward system of the original experiment. ACT-R does not have a mechanism to distinguish recognition from recall, and recall is an all-or-nothing event, thus it was not possible to have a meta-awareness of stimulus familiarity build-up throughout the retrieval process, something which could be leveraged to induce speed-accuracy tradeoffs. This speed-accuracy trade-off may result in relatively faster performance in the low-frequency conditions as participants' threshold to respond may be lower than the model's.

It is fair to argue that our model is substantially more complex than prior efforts, but we argue that this complexity is necessary to understand how fan effects arise from learning. By having our model perform the study equivalently to human participants and by having participants learn associative weights throughout the experiment, we present a model that supports our theory of associative memory and explains how associations are learned and adapt over time.

Acknowledgments

This work was supported by the Office of the Secretary of Defense / Assistant Secretary of Defense for Research and Engineering (LH) and the Office of Naval Research (LH). The views and conclusions contained in this paper do not represent the official policies of the U.S. Navy.

References

Anderson, J. R. (2007) *How Can the Human Mind Occur in the Physical Universe.* Oxford University Press: Oxford.

Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology*, 6, 451-474.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., Qin, Y. (2004). An integrated theory of mind. *Psychological Review, 111*, 1036-1060.

Anderson, J. R., & Reder, L. M. (1999) The fan effect: New results and new theories. *Journal of Experimental Psychology: General*, 128, 186-197.

Anderson, J. R., Bothell, D., Lebiere, C. & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language*, 38, 341-380.

Berry, M. J., & Meister, M. (1998). Refractoriness and neural precision. *The Journal of Neuroscience*, 18 (6), 2200-2211.

Eliasmith, C. (2005). A unified approach to building and controlling spiking attractor networks. *Neural Computation*, 17 (6), 1276-1314.

Hiatt, L. M., & Trafton, J. G. (2015). An Activation-Based Model of Routine Sequence Errors. *Proceedings of the International Conference on Cognitive Modeling.*

Hiatt, L. M., & Trafton, J. G. (2013). The Role of Familiarity, Priming and Perception in Similarity Judgments. *Proceedings of the Conference of the Cognitive Science Society.*

Pyke, A., West, R. L., & LeFevre, J. A. (2007). How Readers Retrieve Referents for Nouns in Real Time: A Memory-based Model of Context Effects on Referent Accessibility. *In Proceedings of the International Conference on Cognitive Modeling.*

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. *Classical Conditioning II: Current Research and Theory, 2,* 64-99.

Thomson, R., Pyke, A., Trafton, J. G., & Hiatt, L. M. (2015). An Account of Associative Learning in Memory Recall. *In Proceedings of the 37th Annual Conference of the Cognitive Science Society.* Austin, TX: Cognitive Science Society.

Thomson, R., Bennati, S., & Lebiere, C. (2014). Extending the Influence of Contextual Information in ACT-R using Buffer Decay. In *Proceedings of the Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Thomson, R., Lebiere, C., Anderson, J. R., & Staszewski, J. (2014). A general instance-based learning framework for studying intuitive decision-making in a cognitive architecture. *Journal of Applied Research in Memory and Cognition Special Issue on Intuitive Decision-Making*.

Thomson, R. & Lebiere, C. (2013a). Constraining Bayesian Inference with Cognitive Architectures: An Updated Associative Learning Mechanism in ACT-R. In *Proceedings of the Conference of the Cognitive Science Society.*

Thomson, R**.** & Lebiere, C. (2013b). A Balanced Hebbian Algorithm for Associative Learning in ACT-R. In *Proceedings of the International Conference on Cognitive Modeling.*

Trafton, J. G., Hiatt, L. M., Harrison, A. M., Tamborello, F. II, Khemlani, S. S., & Schultz, A. C. (2013). ACT-R/E: An embodied cognitive architecture for human-robot interaction. *Journal of Human-Robot Interaction, 2 (1),* 0-55.

Wills, C. L., Cacucci, F., Burgess, N., & O'Keefe, J. (2005). Attractor dynamics in the hippocampal representation of the local environment. *Science,* 308, 873 – 876.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Towards Modeling False Memory with Computational Knowledge Bases

Justin Li (justinnhli@oxy.edu)
Emma Kohanyi (kohanyi@oxy.edu)
Occidental College, 1600 Campus Road
Los Angeles, CA 90041 USA

Abstract

One challenge to creating realistic cognitive models of memory is the inability to account for the vast common sense knowledge of human participants. Large computational knowledge bases such as WordNet and DBpedia may offer a solution to this problem, but may pose other challenges. This paper explores some of these difficulties through a semantic network spreading activation model of the Deese-Roediger-McDermott false memory task. In three experiments, we show that these knowledge bases only capture a subset of human associations, while irrelevant information introduces noise and makes efficient modeling difficult. We conclude that the contents of these knowledge bases must be augmented and, more importantly, that the algorithms must be refined and optimized, before large knowledge bases can be widely used for cognitive modeling.

Keywords: False Memory; Spreading Activation; Knowledge Base.

Introduction

The modeling of human memory phenomena has a long history, from equations describing the strength of individual memory elements over time, to the embedded memory subsystems in modern cognitive architectures. One limitation of memory models, however, is their failure to account for how experimental subjects do not come into the laboratory as a blank slate, but with a large set of common-sense knowledge and facts about the world, as well as associations built up from individual experience. This background knowledge is impossible to fully elicit from subjects and often omitted from computational models. As a result, these models are over-simplified and may fail to account for phenomena in which the contents of memory play a role.

At the same time, the increasing number of artificially intelligent agents that operate in knowledge-rich environments has led to the development of large computational knowledge bases. Knowledge bases such as WordNet (Miller, 1995) and DBpedia (Bizer et al., 2009) endow artificial agents with lexical and conceptual knowledge, allowing them to perform human-like reasoning. These collections of semantic knowledge, in a form that can be incorporated into the long-term memory of cognitive architectures, present an opportunity to build models that match real human memory in scope and scale. Recent work has adapted DBpedia for factual question-answering in the ACT-R architecture (Salvucci, 2015), a task for which the knowledge base is well suited, as it mirrors the use of DBpedia in artificial intelligence research. Whether knowledge bases can be used to model cognitive phenomena outside of reasoning and inference, however, remains an open question.

In this paper, we explore some of the challenges that researchers may face when incorporating large computational knowledge bases into a cognitive model. Specifically, we use WordNet and DBpedia to model the formation of false memories through human associations in the Deese-Roediger-McDermott (DRM) paradigm (Roediger & McDermott, 1995). We selected the false memory task specifically because it involves a broad range of knowledge that large knowledge bases could provide, while requiring associations for which WordNet and DBpedia may not be particularly well suited. The partial success of our model suggests that while large knowledge bases hold promise for general cognitive modeling, they present representational and algorithmic challenges that have yet to be overcome.

Background

The DRM task is a well-known procedure for inducing false memory in humans. Participants are told they are part of a memory experiment and presented with a list of fifteen *stimuli words* at a moderate pace. After the presentation, participants are occupied with a filler task, before being given two minutes to recall as many words from the list as possible. Crucially, the list of words are not random, but are all associated with a *lure*, which itself does not appear on the list. For example, for the lure *"needle"*, the list of words presented to the participants includes *"pin"*, *"sharp"*, *"prick"*, *"haystack"*, *"thorn"*, *"cloth"*. (All words in a DRM list will be in quotes and italicized, with the lure words underlined; all other words will be in quotes but unitalicized.) The result is that experiment participants will recall the lure at roughly the same rate as the stimuli words, and will further report that the lure was presented – a false memory. After a break, another list built around a different lure is presented, for 36 published false memory word lists (Stadler, Roediger, & McDermott, 1999). In the original study, participants recalled 62% of the stimuli words, and falsely recalled the lure 55% of the time.

In a different publication (Roediger, McDermott, & Robinson, 1998), the authors suggested that this phenomenon could be explained through a *spreading activation* mechanism. They hypothesized that the semantic concepts represented by the stimuli words are connected in a *semantic network*; nodes in the network represent concepts, while edges between nodes represent an association of some kind. Thus, every word on a DRM list would be connected to the lure, possibly with additional connections between stimuli words. Each word would also have an *activation value* that represents its salience at any particular time; the higher the activation, the more likely that concept will be recalled at that time. When a stimuli word is presented, it is hypothesized that not only is the activation of that concept *boosted*, but so is the activation

of associated concepts, including the activation of the lure. The presentation of multiple stimuli words would boost the activation of the lure multiple times, causing its activation at the end of the presentation phase to be indistinguishable from the activation values of the stimuli words. Then, during the recall phase, words with the highest activation are recalled. Since participants could not determine whether the high activation of a word is due to its presentation or due to spreading activation (a *source monitoring failure*), they report the lure as having been presented.

Although spreading activation is an intuitive and appealing explanation for how false memories are induced in the DRM paradigm, creating a cognitive model of the task requires capturing human associations between words. The breadth of the stimuli and lure words – which range from everyday objects such as *"window"* and *"pen"* to relatively obscure words such as *"sash"* (a type of window) and *"Cross"* (a pen company) – makes the creation of a comprehensive model challenging. Traditional word-association paradigms cannot cover a sufficiently large range of words, even when converted into a "game with a purpose" and crowd-sourced to players on the internet (Hees, Khamis, Biedert, Abdennadher, & Dengel, 2013).

A previous model of the DRM task estimated word associations from co-occurrence information in a text corpus, using the latent semantic structure to "recall" words that are semantically similar to the stimuli words (Johns & Jones, 2009). As the authors themselves noted, these lexical-semantics techniques only capture the structure of memory at best, but do not shed light on the recall processes. While the resulting model leads to good fits for the stimuli and lure recall rates from the original study, the computational linguistic techniques used were not designed to model recall tasks, requiring a convoluted process for generating the lure. Furthermore, these models cannot accommodate complex reasoning with the encoded concepts, meaning that the knowledge captured by these associations is unusable for modeling human inference.

This paper instead directly tests the original hypothesized spreading activation mechanism, using large computational knowledge bases as the semantic network. The assumption is that the organization of these knowledge bases naturally encode association information, with more strongly associated concepts represented by nodes separated by a shorter network distance. Gleaning association information from computational knowledge bases would be a step towards the ideal of a single source of semantic knowledge that can be broadly used to model both human associations and inference.

Model Description

This section first describes the relevant components of the Soar cognitive architecture, before describing the model built using Soar.

Soar's working memory contains knowledge that is available for immediate reasoning. Working memory is represented as an edge-labeled directed graph, which is matched on and modified by procedural rules. In addition to knowledge in working memory, Soar has a long-term semantic memory, which contains general knowledge about the world. Each piece of knowledge (a node) in either memory is known as a memory element, Knowledge in semantic memory must be retrieved into working memory before it can be used. To do so, a Soar agent must create a cue that describes features of the desired piece of knowledge. Each element in semantic memory is associated with a base-level activation value, which reflects the recency and frequency of the retrieval of the element. The more recently and frequently an element is retrieved, the higher its activation value; however, the activation automatically decays over time. When the agent creates a cue, semantic memory returns the most-activated memory element that matches the cue, and places it in working memory to be matched on by procedural rules.

Spreading activation, as the hypothesized mechanism that leads to false memories, operates on the knowledge in semantic memory. Unfortunately, there is no standardized spreading activation algorithm, nor is there consensus on the meaning of spreading activation. In Soar, every retrieval of a memory element not only boosts the activation of that element, but also boosts the activation of neighboring elements in semantic memory, hence "spreading" the activation (Li & Laird, 2015). The number of elements that receive a boost is implicitly defined by a maximum spreading depth parameter, with a spreading depth of zero meaning that only the retrieved element receives a boost. All neighboring elements (regardless of edge direction) receive the same boost – the effect is not attenuated by distance, nor are there differential effects due to the strength of the connection between elements. In fact, the boost due to spreading is indistinguishable from the boost received by the element retrieved; both changes are to the base-level activations of the elements and will therefore affect future retrievals. This is notably different from the spreading activation in ACT-R, which comes from elements in working memory, is considered separately from the base-level activation of memory elements, and only affects the current retrieval. Since the sources of activation (the stimuli words) are not present (not in working memory) at the time of recall in the DRM task, our model uses Soar's spreading activation mechanism in order to take advantage of its temporal extent.

Agent Description

A Soar agent plays the role of an experimental participate in our model. Before a list is presented, the agent's semantic memory is pre-loaded with the knowledge base for the experiment. The base-level activation of each element is uniform and is not initialized, as there is no consistent method of doing so for all three database. Once the database is loaded, the agent is sequentially presented with the stimuli words as strings. The agent must then retrieve the element that represents the associated concept from semantic memory, causing activation to spread to neighboring elements. Only after this retrieval is the next stimuli word presented, at which

point the agent removes all previous elements from working memory. After all fifteen words from a list have been presented, the agent enters the recall phase. It retrieves the fifteen most activated words (without repetition) from semantic memory, from which the recall statistics are calculated. The semantic memory of the agent, including the activation of the elements, is then reset for the presentation of the next list.

We note two caveats to this agent. First, the base-level activation of each element in the knowledge base is not initialized. Selecting the initial activation is a non-trivial problem. Using the number of connections from each element (Salvucci, 2015) means that activation levels are not consistent between knowledge bases, while using frequency information from a text corpus (Johns & Jones, 2009) may require manually mapping concepts to all their synonyms. For this paper, we do not believe the lack of initialization is the main cause of model error; as we explain in the general discussion, the difficulties do not come from differences in retrieval order, but from whether the correct elements and connections exist in the knowledge base at all We acknowledge, however, that initializing activation is an important part of memory models not captured here, and more exploration into robust algorithms for consistently initialization activation across knowledge bases may be necessary.

The second caveat to our agent is the design of the recall phase. In the human experiments, the participants were given 2 to 2.5 minutes to recall as many words as possible. In contrast, the agent in this model only retrieves the first 15 words, equivalent to a retrieval every eight seconds – a slow but not unreasonable rate. Using ACT-R's simulated retrieval times to approximate the procedural constraints would likely lead to the opposite problem of too many recalled words, since retrievals take less than a second by default (even with additional time for rule firings). Additional memory mechanisms – perhaps rules for determining whether a retrieved word should be reported as a stimuli – may be needed to model the DRM task with higher fidelity.

Metrics

We are interested in two key metrics that were used in the original false memory study:

- The stimuli recall rate, which is the proportion of stimuli words recalled after the presentation of a list, averaged over all 36 lists. The original study reports a stimuli recall rate of 62%, meaning that on average participants recalled 62% of the fifteen words in a list.
- The lure recall rate, which is the proportion of the 36 lists in which the lure was (falsely) recalled. Note that this metric is about a proportion of *lists*, and not about a proportion of the *stimuli words* in a list, and thus has no direct relationship to the stimuli recall rate. The original study reports a lure recall rate of 55%, meaning that on average participants had a false memory of the lure on 55% of the lists.

Before we describe the three experiments with different knowledge bases and their results, we reiterate that the goal of this work is not necessarily to perfectly model the stimuli and lure recall rates. We are not looking for the exact depth limit to spreading activation that should be used in future false memory models. Rather, the experiments below should be seen as an exploration of some of the challenges that cognitive modelers may face when attempting to leverage large knowledge bases, especially on tasks for which the knowledge bases are not designed. Towards this goal, while the metrics above provide a rough sense of the goodness of fit, the discussion for each experiment is more focused on properties of the knowledge base that led to those results.

Experiment 1: Hand-crafted Network

The goal of this experiment is to validate spreading activation as a viable explanation for false memory in the DRM task. The semantic network used in this experiment was created manually from the words in the *"needle"* and *"doctor"* lists. For each list, the fifteen stimuli words are all connected to the lure, with additional connections created based on whether the words are intuitively and informally associated. For example, *"pin"*, *"thimble"*, and *"prick"* are all connected, while none of the three are connected to *"haystack"*. Finally, four connections were added between the stimuli words of the two lists, such as *"injection"* (from the *"needle"* list) and *"medicine"* (from the *"doctor"* list) and *"hurt"* and *"sick"*, for a total of 109 edges between 32 nodes. It is important to note that the resulting network is representative of how semantic networks are depicted in non-computational literature.

Only the *"needle"* and *"doctor"* lists were presented using this network, with an activation decay rate of 0.5 and a spreading depth limit of 1. The results for both lists are similar. The lure is the first word to be retrieved (as it has the highest activation), with the stimuli words for the list retrieved afterwards. As would be expected, since activation spreads only to the immediate neighbors of the stimuli words, the four words that bridge the two lists are also activated, but not the lure of the not-presented list

Although only two lists are used for this experiment, there is no reason to believe that the results would not generalize to similar hand-crafted semantic networks for the other lists. The quantitative results cannot be meaningfully compared to the stimuli and lure recall rates of the original study; however, the qualitative results are in line with the description that the lure is more highly activated than some stimuli words. While there is a tendency for words towards the end of a list to be retrieved first – as would be consistent with the decay of activation over time – the actual order of words retrieved is also affected by the structure of the semantic network due to spreading activation. Since the retrieval order would once again be different if the activation was initialized with other information, the rest of this paper does not consider the order in which words are retrieved. Regardless, this experiment suggests that spreading activation on a naive semantic network could cause the retrieval of the lure, which in this model

indicates the formation of a false memory.

Experiment 2: WordNet

WordNet (Miller, 1995) is a database containing lexical knowledge, and is widely used both independently (for tasks such as parsing and word sense disambiguation) as well as in conjunction with other knowledge bases and ontologies. Nodes in WordNet represent not only words and phrases (for example, "sewing needle"), but also additional information about the meaning of those words, including word meanings (*senses*), synonym sets (*synsets*), antonyms, and certain types of entailments (for example, buying entails paying, so "buy" is connected to "pay"). WordNet nodes that represent words can be identified by an outgoing edge labeled `string`, which links to a string representation of the word; these edges do not exist for other concepts (such as synsets). The version of WordNet imported into Soar's semantic memory contains over 474,000 nodes and 1.7 million edges.

The Soar agent used in this experiment is roughly the same as the one used in the first experiment. The only difference is in the recall phase, when the agent restricts the retrievals to words by specifying the `string` edge in the cue. All words from the DRM lists are used as is, with the exception of *"Bic"* and *"Cross"* from the *"pen"* list. These pen companies do not exist in WordNet and were excluded from the experiment; the *"pen"* list therefore only contains thirteen stimuli words.

For this experiment, separate trials were run for different spreading depths (1 through 6) and different decay rates (0.25, 0.5, 0.75, 0.9).

Results

The overall results are shown in Figure 1. For each parameter setting, we plot both the stimuli recall rate and the lure recall rate, as well as average proportion of recalled words (out of 15) that are neither stimuli words nor the lure (which we shall call *external* words). The human data from the original DRM study is shown for comparison; the results for depths 1 and 2 are left out for reasons explained below. Across all parameter settings shown, the stimuli recall rate ranges from 9% to 41%, well below the reported rate of 62% in humans, while the lure recall rate ranges from 0% to 72%, compared to the reported rate of 55% in humans. In particular, using the ACT-R and Soar default decay rate of 0.5, a spreading depth of 5 results in a lure recall rate of 56%. In general, however, no parameter setting accurately matches human data on both stimuli and lure recall rates.

For spreading depths of 1 and 2, the stimuli words were consistently retrieved, while the lure was never retrieved. Upon examination, this is because WordNet is structured with most words only being connected through word senses and synsets. The node representing *"thorn"*, for example, is connected to three word senses, each of which is connected to a synset – which means that, within a network distance of two, *"thorn"* is not connected to any words at all, never mind the lure *"needle"*. Since the retrieval cue used by the agent limits results to words, the retrieval fails after the stimuli words are retrieved. This explains both the high stimuli recall rate and why the lure is never retrieved.

The data shows additional trends regarding the stimuli and lure recall rates. In general, the spreading depth is *proportional* to the lure recall rate but *inversely proportional* to the stimuli recall rate. That is, the stimuli recall rate decreases as spreading activation extends deeper from the stimuli word, while the lure recall rate increases from the same manipulation.

These results can be explained by the same WordNet structure mentioned previously. When spreading activation is limited to nearby nodes, only a small number of words (as opposed to word senses, synsets, etc.) are boosted, hence the majority of words retrieved are the stimuli. When the depth limit is increased, however, spreading activation now reaches other words in the synsets. These words – which may include the lure – may in fact receive activation boosts spread from multiple stimuli words. The word "shot" falls into this category, as it means both *"injection"* and *"hurt"* (as in *a solid shot to the chin*). Other external words may simply be boosted by stimuli words later in the list, and therefore have higher activation during the recall phase than stimuli words earlier in the list. Together, this leads to a decrease in the stimuli recall rate as well as an increase in the lure recall rate.

Discussion

Although the lure recall rate from WordNet spans a range that includes the human lure recall rate of 55%, the structure and content of WordNet does not directly match human associations. The nodes representing the stimuli words in WordNet are not structured such that activation will spread to the lure. We discuss two categories of such failure here: cases where additional edges lead to model errors, and cases where edges are missing.

First, as we noted, WordNet is structured with individual words arranged in "spokes" around lexical constructs such as synsets. While synsets do represent some of the relationships between stimuli words and the lure – as in *"syringe"* and *"needle"* – they are not the only relationships around which words are organized. Since WordNet is a dictionary in knowledge base form, it also contains information about the derived form of words, such as the relationship between *"inject"* and the words "injectable", "injecting", "injection", and "injector". With the exception of *"sit"* and *"sitting"* in the *"chair"* list, derived words do not appear in the DRM lists, and more importantly, are unlikely to be produced during human recall. This mismatch may be due to the *lexical* relationships encoded in WordNet, as opposed to the *conceptual* relationships on which spreading activation is hypothesized to occur. Human participants would only produce one word for each concept, but spreading activation (at least over WordNet) leads to the retrieval of multiple derived words. Algorithmic changes may be necessary before spreading activation can correctly model the generation of false memory; we propose one such change in the general discussion.

Although WordNet contains connections that extend beyond

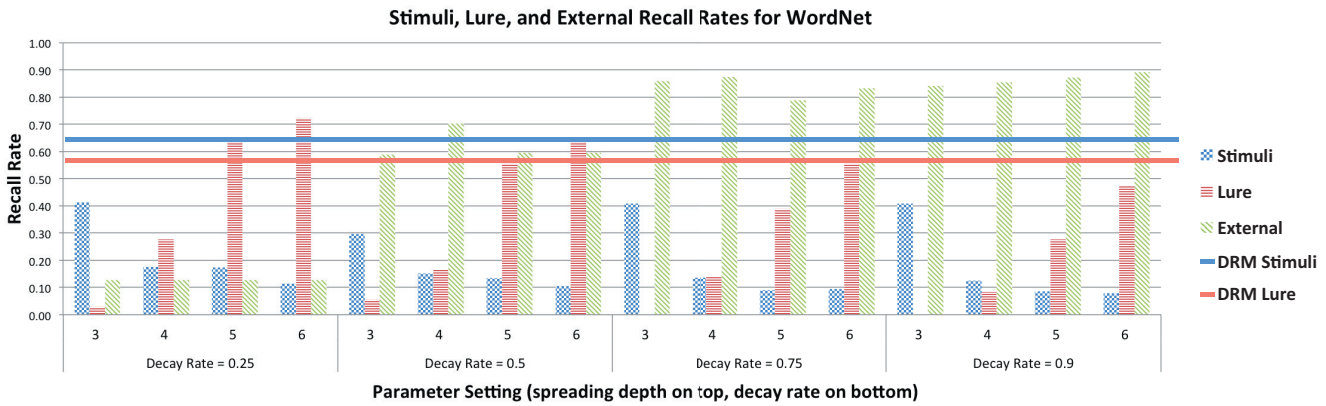


Figure 1: Results from using WordNet as the knowledge base.

human associations, it fails to capture other relationships that the DRM lists exploit. A careful examination of the word lists reveals that they contain multiple types of associations. Some, such as antonyms (*"high"* and *"low"*), are encoded in WordNet despite being more conceptual. Others, however, are not captured despite being lexical in nature. For example, the *"high"* list contains the word *"noon"*, clearly intending to invoke the phrase "high noon". Crucially, while "high noon" does exist as a phrase in WordNet, it is not connected to its component words *"high"* and *"noon"*. At the same time, other idiomatic phrases, such as "*needle* in a *haystack*" and "making a *mountain* out of a *molehill*", are not represented in WordNet. Also missing are cultural references; the inclusion of *"tiger"* and *"bear"* in the *"lion"* list appears peculiar, but may be explained by the lyric *lions and tigers and bears, oh my!* from *The Wizard of Oz*. Unlike the first type of failure due to an over-abundance of connections, there is no algorithmic solution to missing data, at least not without expanding the database using a text corpus, which presents challenges of its own.

Mismatched and missing data is not unexpected in large knowledge bases, although in this case some of them seem to arise from WordNet's specialization in lexical knowledge. Our third experiment looks at whether a different knowledge base may lead to a better model of human associations in false memory.

Experiment 3: DBpedia

DBpedia (Bizer et al., 2009) is a knowledge base created using information from the online encyclopedia Wikipedia. The nodes in DBpedia represent articles on Wikipedia (or more accurately, they represent the concepts that the Wikipedia articles describe), while the edges come from the categories to which the articles belong, as well as the *infoboxes* that provide basic information. As a result, the type and amount of information varies between concepts. The version of DBpedia used in this experiment contains 6 million nodes and 27 million edges.

The size and scope of DBpedia led to two differences in this experiment from the previous ones. First, since DBpedia does not contain a comprehensive dictionary of English words, and not all words in the DRM lists have their own Wikipedia article, the stimuli words can no longer be presented as strings. Instead, we manually mapped each word to a concept in DBpedia, mostly following the redirections on Wikipedia. This led to some words being mapped onto the same concept (*"waste"* and *"refuse"* both mapped onto "waste"), while others mapped onto concepts that are overly specific (*"garbage"* mapped onto "municipal solid waste"). More problematic were words that differed in meaning from their Wikipedia articles. Words from the *"thief"* list are good examples: Wikipedia does not contain articles for *"thief"*, *"robber"*, *"burglar"*, *"bandit"*, or *"criminal"*, only articles for "thievery", "robbery", "burglary", "banditry", and "crime". These words were excluded from this experiment.

To accommodate the size of DBpedia, a custom Python script that simulated spreading activation was used instead of Soar, although the same algorithm as Soar's semantic memory is followed. For this experiment, the fifteen "retrieved" concepts are simply the fifteen most-activated nodes. The size of DBpedia and the density of its connections remains daunting; as an example, a fifth of the nodes in DBpedia are only two connections away from the nodes selected for the *"army"* list. This makes spreading beyond a depth of 2 untenable. As a result of these two problems, only about half the lists (seventeen) were used in this experiment, with an average of 14.1 concepts.

Results

Due to the reduced dataset, the results in this section should be treated with some skepticism; however, we believe they are nonetheless representative of using DBpedia to model false memory and human associations.

For spreading depth 1 at the default decay rate of 0.5, spreading activation on DBpedia resulted in stimuli and lure recall rates of 15% and 0% respectively; for spreading depth 2, the stimuli recall rate decreases to 3%, while the lure recall rate increases to 12%. These numbers follow the trends found from the WordNet experiment. To understand the low lure

recall rate, we found it instructive to look at the *"shirt"* list, one of two lists for which the lure was consistently retrieved. Unlike other DRM lists, the *"shirt"* list is unique in that the vast majority of items belong to the same category. This shared classification means that the lure is only a network distance of two away from the stimuli words, and is therefore sufficiently boosted in activation for it to be retrieved. In contrast, the stimuli words for other DRM lists do not conform as neatly to the taxonomic structure of DBpedia – the lure is not as directly connected to the stimuli, causing the lure to not be retrieved.

That the lure is not retrieved, however, does not mean that the stimuli words are retrieved; the highly connected network structure also led to the low stimuli recall rate. Page links on the internet are known to have a small-world structure, where the pairwise distance between all nodes are small and where there are many nodes with large degrees. For example, *"anger"* is connected to "red", which in turn is connected to over 600 concepts, mostly organizations whose representational colors include red. Because these "hub" nodes are often connected to multiple stimuli words, their activation is boosted above that of the stimuli words and are retrieved instead, resulting in a low stimuli recall rate.

Discussion

The failures in both WordNet and DBpedia are representational; we discuss these issues in the next section. For DBpedia alone, we faced the additional difficulty of mapping the stimuli and lure words to a concept. One concern not yet raised is that the choice of concepts used to represent nodes requires association and reasoning on the part of the modeler. A number of words in the DRM lists are polysemous; *"prick"* and *"hurt"*, for example, would fit just as well as "goad" and "heckle" into a different *"needle"* list (as a verb instead of as a noun). If DBpedia is to be used for modeling associations and false memory, a better protocol would be for unknowing coders to determine which concepts correspond to the lure and the stimuli words. This would remove confirmation bias that may be inherent in how words are currently mapped to concepts.

General Discussion

This paper attempted to use large computational knowledge bases to model the human associations that lead to false memory in the DRM paradigm. Our model was able to qualitatively recreate the DRM false memory phenomenon, but only on a hand-crafted semantic network that resembles their traditional depiction. When large computational knowledge bases such as WordNet and DBpedia are used, however, the naive spreading activation algorithm fails to simultaneously match the stimuli and lure recall rates. We believe that these results are indicative of three general problems with using large knowledge bases in cognitive modeling: missing data from the knowledge base, missing connections between existing data, and finally, the sheer amount of existing data.

Of the three, the missing data problem is the hardest to solve. The type of common sense knowledge required to make associations in the DRM task is neither lexical nor conceptual – it exists neither in a dictionary nor in an encyclopedia. One example of such knowledge is the fact that *"rubber"* is *"elastic"*, *"springy"*, *"flexible"*, and *"resilient"*. It is infeasible to manually encode all descriptions for all objects, and it may be necessary to employ techniques from information retrieval and natural language processing to extract this knowledge from text.

Even for concepts/words that exist in the knowledge base, neither WordNet nor DBpedia fully capture the relationships between their nodes. Some of these missing relationships, such as phrases from popular culture, can only be obtained through similar means as the missing concepts/words; others, by systematically adding edges to these knowledge bases, such as connecting phrases to their component words. Perhaps more relevant for cognitive modelers, however, is that there is no consensus on the cognitive plausibility of the content and structure of knowledge bases. In understanding the experimental results of this paper, we have tried to determine how the stimuli words relate to the lure, and whether these relationships generally apply to other concepts. A complete catalog of human associations would more clearly indicate the types of connections that knowledge bases currently lack.

The final problem of the scale of the data is only made worse by the addition of missing knowledge. The solution here may be more algorithmic in nature, by modifying the spreading activation algorithm such that it remains valid as the size of the knowledge base grows. One possibility is for spreading to occur only on particular edges, perhaps informed by the context of the retrieval. This is similar to using theory to extract a smaller, more specialized network on which the network distance may be more meaningful (Tenenbaum, Griffiths, & Kemp, 2006). Such an algorithm would reduce the computational requirements of spreading activation, while simultaneously filtering out connections that are irrelevant for fitting human data. The same mechanism may also allow lexical, conceptual, and other knowledge to exist in the same knowledge base, as a unified semantic memory to be used in cognitive modeling, without leading to the confusions demonstrated in the results of this paper.

With more refined algorithms that can efficiently operate on millions of concepts and relations, large computational knowledge bases can become a valuable resource for modeling the wealth of background knowledge that participants bring into experiments.

References

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., et al. (2009). DBpedia — a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, *7*(3), 154–165.

Hees, J., Khamis, M., Biedert, R., Abdennadher, S., & Dengel, A. (2013). Collecting links between entities ranked by human association strengths. In *Proceedings of the 10th European semantic web conference.*

Johns, B. T., & Jones, M. N. (2009). Simulating False Recall

as an Integration of Semantic Search and Recognition. In *Proceedings of the 31st annual conference of the Cognitive Science Society (CogSci).*

Li, J., & Laird, J. E. (2015). Spontaneous retrieval from long-term memory for a cognitive architecture. In *Proceedings of the 29th AAAI conference on artificial intelligence (AAAI).*

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, *38*(11), 39–41.

Roediger, H. L., & McDermott, K. B. (1995). Creating false memories: Remembering words not presented in lists. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *21*(4), 803–814.

Roediger, H. L., McDermott, K. B., & Robinson, K. J. (1998). The Role of Associative Processes in Producing False Remembering. In M. A. Conway, S. E. Gathercole, & C. Cornoldi (Eds.), *Theories of Memory II.*

Salvucci, D. D. (2015). Endowing a Cognitive Architecture with World Knowledge. In *Proceedings of the 37th annual conference of the Cognitive Science Society (CogSci).*

Stadler, M. A., Roediger, H. L., & McDermott, K. B. (1999). Norms for word lists that create false memories. *Memory and Cognition*, *27*(3), 494–500.

Tenenbaum, J. B., Griffiths, T. L., & Kemp, C. (2006). Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, *10*(7), 309–318.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Using behavior to decode allocation of attention in context dependent decision making

Michael Shvartsman (ms44@princeton.edu)
Princeton Neuroscience Institute, Washington Rd., Princeton, NJ 08544 USA

Vaibhav Srivastava (vaibhavs@princeton.edu)
Department of Mechanical and Aerospace Engineering, 41 Olden St., Princeton, NJ 08544 USA

Narayanan Sundaram (narayanan.sundaram@intel.com)
Intel Parallel Computing Lab, 2200 Mission College Blvd., Santa Clara, CA 95054 USA

Jonathan D. Cohen (jdc@princeton.edu)
Princeton Neuroscience Institute, Washington Rd., Princeton, NJ 08544 USA

Abstract

We present a model of the dynamics of adaptive attention allocation in the AX Continuous Performance Test (AX-CPT), a simple context dependent decision making task of interest to the research communities concerned with cognitive control, schizophrenia, anxiety and aging (Braver et al., 2001; Cohen et al., 1999; Eysenck et al., 2007). We ground it in our recent theory of decision making under dynamic context, that assumes humans use sequential Bayesian inference to combine information from multiple sources in perception and (optionally) memory over time. The theory generalizes the well-known diffusion decision model of single-stimulus decision making (DDM; Ratcliff, 1978). Our first result is a new analysis that shows how memory encoding and retention can yield a variable initial condition for either a single- or multiple-stimulus decision, providing a theoretical grounding to the assumption of variability in initial condition previously shown to improve data fits for the DDM. Our second result is using this model to decode attention allocation from behavioral data in a novel quantitative payoff manipulation in the AX-CPT, showing that our model can capture the differences in how subjects encode and retrieve contextual information when the relative emphasis on task speed and accuracy is changed.

Keywords: decision making; context processing; sequential inference; Bayesian inference; wiener process; ornstein-uhlenbeck process;

Introduction

If there is one thing that defines human behavior, it is its pervasive adaptive nature. Such a viewpoint is perhaps best underpinned by the seminal work of Swets and colleagues on signal detection (e.g Tanner and Swets, 1954), who showed that even the simple task of detecting a flash of light in a uniform background is amenable to strategic variation that trades off correct responses against false alarms in a way sensitive to reward and the statistics of the environment. The Signal Detection Theory model of this behavior assumed observers performed the equivalent of a fixed-sample likelihood ratio test when asked to respond, and was soon generalized to the case of variable sample size (Edwards, 1965; Laming, 1968; Stone, 1960), and to continuous time as the well-known Diffusion Decision Model (DDM; Ratcliff, 1978).

These models naturally capture a second tradeoff in decision making: that of speed against accuracy, and find support from a wide range of behavioral and neural data (e.g. Bogacz et al., 2006; Bogacz and Gurney, 2007; Gold and Shadlen, 2007; Kira et al., 2015; Ratcliff et al., 2004; Turner et al., 2015; van Vugt et al., 2012).

We consider the above models to be *fixed context* models because they treat context, which we operationalize as additional task-relevant information, as fixed/known over the course of the decision. Our theory (Shvartsman et al., 2015) removes the assumption that the context is known, and considers the dynamics of processing the context simultaneously with an additional stimulus we term the decision *target*. The theory can be stated both as a sequential Bayesian inference model, and (in the continuum limit) as a nonlinear diffusion model, and is therefore a formal generalization of the models discussed above, as well as a previous Bayesian model of the Flanker task by Yu et al. (2009). In prior work, we used the same parameters from Yu and colleagues' paper to generate plausible behavior patterns in the AX Continuous Performance Task (AX-CPT), a task that differs from the Flanker task both in the time of stimulus presentation, and in response rules. The AX-CPT, which we describe in detail below, is arguably the simplest task where participants are required to combine information seen during different disjoint time intervals to make a decision, and is our focus in this paper.

When deciders include additional contextual information in their decisions, a third tradeoff naturally arises, that of allocating processing (attentional or otherwise) between the available sources of information. In this paper we explicitly consider this tradeoff. In addition to being a formal generalization to the work on fixed-context decision making discussed above, it also bears some relation to work on attention allocation under multi-sensory integration (e.g. Sheppard et al., 2013), which it extends considering stimuli in memory. It is also complementary to work on the perception-memory tradeoff on longer timescales in the prospective memory paradigm (Einstein and McDaniel, 2005), and mechanistic work combining multiple sequential samplers in the ACT-R cognitive architecture (van Maanen et al., 2012).

Our main contributions are as follows: first, we provide additional insight into of our theory of decision making under dynamic context presented in Shvartsman et al. (2015)

by providing analytical expressions for the posterior distribution over context after memory encoding and retention, and thereby provide a new motivation for the assumption of initial condition variability previously argued to be necessary to fit human data in the fixed-context DDM (e.g. Ratcliff and Rouder, 1998). Second, we apply the model to a dataset of humans performing the AX-CPT under a novel quantitative payoff manipulation, and use the model to develop a quantitative understanding of how participant strategy changes with task goals.

An overview of the theory and application to the AX-CPT

We assume that decision making can be understood as a sequential Bayesian inference process, specifically that the agent uses sequentially-drawn samples from external input and/or memory to compute the joint posterior probability over the identity of some true *context* and decision *target* over time. The agent maps from this joint posterior to a response probability using a known response rule, and uses a fixed threshold defined over the response probability to stop sampling and respond. We make a crucial distinction between our *theory* of decision making and individual *task models* that can be derived from the theory by applying it to specific tasks, as we do here. Our previous work has shown how our formalism can be used to derive different models when the additional context stimulus is either in perception or memory, under different response mappings (Shvartsman et al., 2015).

In this paper we focus on the AX Continuous Performance test (Servan-Schreiber et al., 1998), arguably the simplest decision making task that conditions responses jointly and uniquely on a perceptual and memory stimulus. The task is illustrated in Fig. 1: participants see one of two context stimuli (by convention labeled 'A' or 'B') followed by one of two targets ('X' or 'Y'), and make one response (e.g. 'left') to AX and BY pairs, and the other (e.g. 'right') to AY and BX pairs.

Formally, we assume the agent conditions a decision based on her posterior belief over the identity of some unknown true *context* and some true *target*. We denote by C, G random variables representing the possible draws of context and target, and $r(\cdot)$ a function from the distribution $P(C,G)$ to a distribution over responses, which for the AX-CPT is the exclusive-or function:

$$r(P(C,G)) = \begin{cases} r_0, \text{ with} & P(C=\text{'A'}, G=\text{'X'}) + \\ & P(C=\text{'B'}, G=\text{'Y'}) \\ r_1, \text{ with} & P(C=\text{'A'}, G=\text{'Y'}) + \\ & P(C=\text{'B'}, G=\text{'X'}). \end{cases} \quad (1)$$

The agent receives evidence samples s^C and s^G drawn i.i.d. from the environment, and updates her posterior distribution over the pair (C,G) using Bayes' rule. We denote by t_{c}^{on} the time at which the context appears and $t_{\text{c}}^{\text{off}}$ the time at which it disappears, and likewise $t_{\text{g}}^{\text{on}}, t_{\text{g}}^{\text{off}}$ for the target. This implies

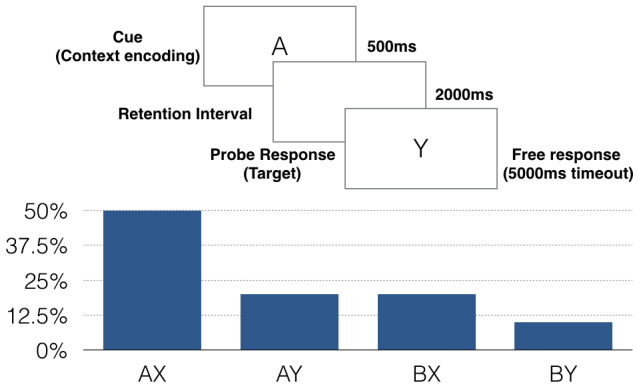


Figure 1: *Top*: sequence diagram for one trial of the AX Continuous Performance Test. *Bottom*: frequency of each trial type in the experiment. Actual letters seen were randomized for each subject.

the following update at timestep τ, assuming independence of the context and target evidence streams:

$$P_\tau(C,G \mid s^C, s^G) \propto P(s^C \mid C) P(s^G \mid G) P_{\tau-1}(C,G). \quad (2)$$

Before the target appears, the target likelihood is uniform over all the targets. Because there are only two responses (i.e. hypotheses), we can rewrite this update as a likelihood ratio update, and convert it into a nonlinear transformation of two diffusion processes. We refer the reader to the supplement of (Shvartsman et al., 2015) for the full derivation, and here only give the final expression:

$$\log Z = \log \frac{P_0(C=c_0, G=g_0) e^{z_c^t} e^{z_g^t} + P_0(C=c_1, G=g_1)}{P_0(C=c_0, G=g_1) e^{z_c^t} + P_0(C=c_1, G=g_0) e^{z_g^t}}. \quad (3)$$

Here, the target particle z_g^t is stationary until the target appears (t_{g}^{on}) and then evolves according to a Wiener process with drift:

$$dz_g^t = a_g dt + \sigma_g dW. \quad (4)$$

The context particle z_c^t evolves according to a Wiener process with drift from when it appears (t_{c}^{on}) until it disappears. Once the context disappears from perception, the memory system can provide continued samples $\hat{s}^C$ after the stimulus goes away, but with some constant probability d at each time step it can start to provide uninformative noisy samples s^U. Assuming the agent has a good estimate of d, she also knows that the probability of receiving an informative sample exponentially decays with time, such that the time-varying likelihood distribution is the following mixture:

$$f(s^C) = (1 - \exp(-\lambda t)) f(\hat{s}^C) + \exp(-\lambda t) f(s^U) \quad (5)$$

, where $f(\cdot)$ refers to the density of its argument. That is, the informative component of the likelihood decays – as does its

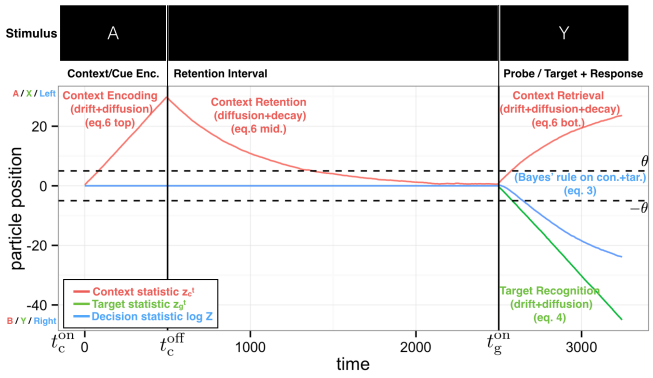


Figure 2: Average dynamics of the CDDM model for the AX-CPT. The ordinate does triple-duty in denoting the position of all three particles in their own spaces. In this figure, context almost (but not quite) decayed fully during the retention interval.

variance, if we make the standard assumption that evidence distributions are Gaussian. We model this retention and forgetting process in continuous time as a zero-mean Ornstein-Uhlenbeck (O-U) process, which has the same exponential decay property, though we leave the derivation of an explicit connection to future work. Finally, we model the retrieval process by switching the O-U process to a nonzero-mean process at t_g^{on}. Our O-U assumption could be alternatively motivated by assuming the memory system is implemented using a leaky competing accumulator model with large leak and mutual inhibition (Bogacz et al., 2006). The full expression for the context particle dynamics is as follows:

$$dz_c^t = \begin{cases} a_e dt + \sigma_c dW & \text{if } t_c^{on} \le t \le t_c^{off} \\ -\lambda z_c^t + \sigma_c dW & \text{if } t_c^{off} \le t \le t_g^{on} \\ -\lambda z_c^t + a_c dt + \sigma_c dW & \text{if } t_g^{on} \le t \end{cases}. \quad (6)$$

We assume by convention that $\sigma_g = \sigma_c = 1$ in order to make the model identifiable (but cf. Bitzer and Kiebel, 2015, for possible consequences of this choice). We therefore omit the Wiener noise coefficients in the notation that follows. Fig. 2 shows the average particle dynamics for the model. The full set of model parameters and their interpretation is given in Tab. 1 (we discuss parameters not related to the decision process itself later in the paper).

We rely on a number of properties of the AX-CPT that enable the analysis that follows (some of which are shared by other tasks as well). First, the context presentation and retention interval durations are both independent of the subject's actions and usually set a priori. Second, responses are only allowed when the target is on the screen. Given those two properties, the context particle distribution at the time of target appearance can be written in closed form, as follows.

First, define $\Delta t_e \triangleq t_c^{off} - t_c^{on}$ and $\Delta t_r \triangleq t_g^{on} - t_c^{off}$. Next, let z_c^e denote the position of the context particle at the end of encoding (t_c^{off}), and z_c^r denote the position of the particle when the target appears and retrieval begins (t_g^{on}). We can write $\mathbb{E}[z_c^r]$, explicitly, recalling that it evolves as an O-U process, and that therefore its distribution is $\mathcal{N}\left(z_e^c e^{-\lambda\Delta t_r}, \frac{1}{-2\lambda}(e^{-2\lambda\Delta t_r} - 1)\right)$[1]. We take care to use the law of total expectation, since z_e^c is itself a random variable with distribution $\mathcal{N}(a_e\Delta t_e, \Delta t_e)$.

$$\mathbb{E}[z_c^r] = \mathbb{E}[\mathbb{E}[z_c^r \mid z_e^c]] = \mathbb{E}\left[z_e^c e^{-\lambda\Delta t_r}\right] = a_e\Delta t_e e^{-\lambda\Delta t_r}. \quad (7)$$

We do the same for the variance, using the law of total variance:

$$\mathrm{Var}[z_c^r] = \mathbb{E}_{z_e^c}[\mathrm{Var}[z_c^r \mid z_e^c]] + \mathrm{Var}_{z_e^c}[\mathbb{E}[z_c^r \mid z_e^c]] \quad (8)$$

$$= \frac{1}{-2\lambda}(e^{-2\lambda\Delta t_r} - 1) + \Delta t_e e^{-\lambda\Delta t_r}. \quad (9)$$

In the case where context is not retrieved from memory once the target appears, the resultant model is a DDM with initial condition variability (Ratcliff and Rouder, 1998), previously used to fit fast errors in decision making. Our analysis here formally provides a psychological interpretation for such fast errors: specifically, it suggests that data that is better fit using sampling or diffusion models with a variable initial condition may reflect subjects' memory of relevant contextual information. Furthermore, it provides an argument on theoretical grounds for using Gaussian rather than the uniform initial condition variability (e.g. Ratcliff and McKoon, 2008), especially if subjects may be using contextual information. Such a model may predict instantaneous decisions ('ultra-fast guesses') for some parameter settings that reflect strong memory encodings, which may be smoothed out to a multi-modal response distribution if nondecision time variability is included in the model.

A particularly elegant property of the model is that each component has both distinct psychological interpretation, and distinct effects on response time distributions. θ reflects the speed-accuracy tradeoff and therefore affects the shape of the RT distributions and the error proportion across all trial types. The three context parameters affect different portions of the RT distribution: a_e reflects encoding of the context, and contributes primarily to the early portion of the RT distributions due to how it affects the initial condition for the decision; λ reflects memory retention and contributes both to early and late portion of RT distributions because it persists both during retention and retrieval; and a_c reflects memory retrieval and primarily contributes to middle and late portions of the RT distributions because retrieval begins at target onset and saturates over time. All three context parameters will identically affect trial types that share a context (e.g. AX and AY). On the other hand, a_g reflects perceptual processing of the target stimulus and therefore affects the RT distribution throughout, but identically affects same-target pairs (e.g. AX and BX)

Thus, while differences between different trial types help uncover differences between context and target processing writ large, differences between different portions of the RT

[1] Note that we use the $\mathcal{N}(\mu, \sigma^2)$ parameterization.

distribution may index differences in subcomponents of context processing. For example, high a_e will increase the number of very fast correct responses when the context predicts the correct target (e.g. AX) but also contribute fast errors when it does not (e.g. AY). In this way, the model may provide a more precise characterization of context processing differences than the behavioral indices in primary use today (e.g. Braver, 2012).

Parameter	Interpretation	Prior range
a_e	Context encoding drift	0 to 5
a_c	Context retrieval drift	0 to 5
a_g	Target recognition drift	0 to 5
λ	Memory decay	-0.05 to 0.2
θ	Decision threshold	0 to 20
μ_{t_0}	Mean of non-decision time	0 to 1000ms
σ_{t_0}	Standard deviation of non-decision time	0 to 200ms
p_c	Probability of contaminant RT	0 to 1
$P_0(\cdot)$	Prior distribution over trial types	Set to true trial distribution

Table 1: Model parameters, interpretation, and prior ranges.

A payoff adaptation experiment

To test the ability of our model to precisely estimate attention allocation from behavioral data, we gathered a dataset of humans performing the AX-CPT task on Amazon's Mechanical Turk. Mechanical Turk is a web-based marketplace for "human intelligence tasks", short web-based tasks that pay small sums of money. It is emerging as a standard method for high-throughput collection of data for psychological experiments (Crump et al., 2013). Our experiment was designed using custom front-end code, using psiTurk (McDonnell et al., 2015) for back-end and interfacing with Amazon. Response times were collected with the JavaScript high resolution timer API, which theoretically promises at least millisecond-level accuracy. Experiment code is available at `https://github.com/mshvartsman/axcpt-psiturk-coffeescript`.

We searched the parameter space of a preliminary version of the model to devise four different payoff schemes intended to differently prioritize speed and accuracy, illustrated in Tab. 2. The first two were designed to encourage participants to be particularly fast or accurate relative to each other. The other two were designed to encourage intermediate behavior, with either high or low overall cost.

Each subject started the experiment by reading the instructions, and then practicing the two A and B contexts separately. To test that they have learned the task, they next had to complete 10 correct trials in a row. If they failed to do so after 50 trials have passed, they were ejected from the experiment. If they succeeded, they completed 240 trials of the AX-CPT divided into 24 blocks. They were told their speed, accuracy, and points gained after every trial, and a running total every block. They earned $1 for participating, plus $1 for each 1000 points they earned in the experiment. We collected 20 subjects per condition, and most earned between $3.50 and $4.50.

Cond.	Error cost	Time cost (/s)	Intent
(A)	10	2	Prefer accuracy
(B)	2	10	Prefer speed
(C)	10	10	Balanced (high cost)
(D)	2	2	Balanced (low cost)

Table 2: Quantitative payoffs given to the participants. Correct responses were always worth 20 points. Both correct and error responses are penalized for time.

Each participant's actual letters were randomized, but in the remainder of the paper we label them as A, B, X, and Y according to their frequencies: each set of 240 trials contained 50% (120) AX trials, 20% (48) AY and BX trials, and 10% (12) BY trials, randomly distributed throughout the experiment. This is conventional in AX-CPT designs, and is what allows the AX-CPT to behaviorally index the allocation of attention. The reasoning is as follows: AY and BX trials have the same joint probability, but different probabilities conditioned on either knowing the true context (A or B) or target (X or Y). Therefore, any asymmetry between these two trials is argued to reflect differences in processing the two stimuli: specifically, better AY performance suggests more attention to the context, and better BX performance suggests more attention to the target (but cf. Lositsky et al. 2015 for evidence that this index may be overly simplistic).

Fig. 3 shows basic behavior summaries in the speed and accuracy conditions: the manipulation was successful in encouraging subjects to adjust their speed and accuracy, as evidenced by the mean behavior. Of more interest is that the effect was not uniform across the four trial types. When pressured to respond more quickly while sacrificing accuracy, subjects sacrifice accuracy in B trials more than in A trials, with the speedup primarily seen in BY, the rarest trial type. The two balanced conditions (not shown in the figure) for the most part showed intermediate behavior, with the high-cost condition patterning closer to the speed-encouraging condition, and the low-cost balanced condition patterning with the accuracy-encouraging condition. We focus the remainder of the analysis on the two extreme conditions, where the patterns are clearest.

Model fitting details

While first passage times for diffusion and O-U processes are known, the nonlinear transformation that forms our model's decision variable makes these analytics inapplicable. Therefore, we derive best fits by simulation, using the EP-ABC algorithm (Barthelmé and Chopin, 2014, as implemented at `https://github.com/sbitzer/pyEPABC`). It bears men-

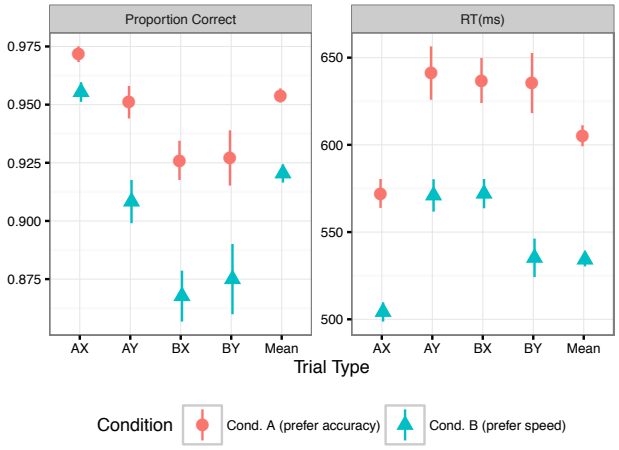


Figure 3: Mean accuracies and response times for the subjects across the four trial types and our two conditions of focus, showing slower speed and higher accuracy overall in the accuracy-focused condition, but also that these changes do not affect all trial types equally. Refer to Tab. 2 for payoff conditions. Error bars are standard errors.

tioning that our use of Bayesian methods for fitting is entirely orthogonal to our theoretical assertion that decision making in humans proceeds by Bayesian inference.

The original EP algorithm (Minka, 2001) approximates the joint posterior density of model parameters by a multivariate Gaussian, converting the Bayesian inference problem to an iterative optimization problem. Approximate Bayesian Computation (ABC) methods contend with performing Bayesian inference when simulation is possible but no likelihood is available, usually by assuming that parameters under which the simulator generates data 'similar enough' to the real data have high likelihood. EP-ABC as we apply it here assumes individual RTs are drawn i.i.d., which allows us to define 'similar enough' to mean 'generates each individual data point to within 1ms'. We used EP-ABC with 5 passes over the dataset and a minimum of 3000 accepted samples per human data point, with EP hyper-parameter α set to 0.3.

The model as described above uses 5 parameters to describe the decision process (three drifts, one decay, one threshold). To describe actual response times, we add two additional assumptions. First, that there is a normally distributed offset to the decision time – a 'non-decision time' reflecting early perceptual processing, motor planning, and the motor response itself. This adds two parameters (mean and standard deviation of the non-decision time). Second, we assume that with some probability RTs are generated not from our model, but from a $\mathcal{U}(0,5s)$ distribution of 'contaminant' RTs. We estimate the proportion of contaminant RTs from the data, adding another parameter. These assumptions are standard in parameter fitting for the fixed-context DDM (e.g. Ratcliff and Rouder, 1998) and provide a probability floor that makes it easier for EP-ABC to handle some unusually fast or slow RTs in our dataset. We used proper uniform priors over plausible parameter ranges for all the parameters. Tab. 1 lists all of the parameters and their prior ranges.

To understand condition-level differences, we generated 10000 samples from the multivariate posterior density for each subject, and produced average parameter estimates for each condition based on these samples. This heuristic weights subjects' posterior means in proportion to their covariances better than simple averaging of the means as point estimates, and without the ideal fully-hierarchical treatment that is extremely challenging in our setting. Because there is stochasticity in both the simulator and the fitting method, we repeated the model-based analysis twice. While there was some variability in the parameters estimated for each subject, the signs of the difference between parameters for the speed and accuracy condition was the same across both runs for all parameters, so we focus our interpretation on those differences. Model code is available at `https://github.com/mshvartsman/cddm`.

Results

Tab. 3 shows the combined parameter estimates. We remark on a number of properties: first, the contaminant proportion is low, and similar across the fits. Second, the threshold is higher in the accuracy-focused than in the speed-focused condition, consistent with the idea that it governs the speed-accuracy tradeoff. Non-decision times are shorter and less variable in the accuracy-focused than in the speed-focused condition – perhaps an indicator of greater focus overall. This point is also supported by the fact that the accuracy condition shows a higher total drift (summed across the three drift variables) than the speed condition, and by the fact that it shows slightly less memory decay.

Parameter	Accuracy condition	Speed condition
a_e	3.1701	2.4623
a_c	1.0576	0.8088
a_g	0.7648	1.1681
λ	0.0756	0.0834
θ	10.9656	6.1589
μ_{t_0}	406.2522	435.1687
σ_{t_0}	76.7926	94.8328
p_c	0.0513	0.0530

Table 3: Fit parameter values in aggregate across the two subject groups. See text for fit details.

Most interesting, however, is how this drift is allocated. In the accuracy-focused group, both context-involving drifts (encoding a_e and retrieval a_c) are higher than the corresponding drifts in the speed-focused group, and the pattern is reversed in the target drift a_g. That is, when incentivized to be more accurate, participants rely more on their memory and contextual information and slightly less on the perceptual stimulus in front of them. Since the task is designed such that context and target information is equally useful in being

able to make the correct response, we suspect that this has to do with the effort involved in encoding, maintaining, and reinstating the memory of the contextual rule – something that is probably worth doing more of when accuracy is more important.

Discussion and Conclusions

In this work, we applied our theory of decision making under dynamic context to the AX Continuous Performance Test. First, we showed how the simple memory encoding and retention dynamics of the model map to variability in the initial condition of a diffusion decision process, providing both further theoretical grounding for the use of variable initial conditions in data fits, and a better understanding for the psychological origin of fast errors in decision making.

We next applied our model to estimate allocation of attention between perception and memory in a novel quantitative payoff manipulation in the task as measured on Amazon's mechanical turk. This manipulation succeeded in not only changing participants' speed-accuracy tradeoff, but also their attention allocation tradeoff between perception and memory. The ability to manipulate attention allocation continuously rather than using discrete task dimensions (e.g. deadlines, distractors) as used previously may pave the way to more quantitative mapping out of the attention allocation strategy space.

Finally, our method of measuring the relative allocation of attention between the cue and probe stimulus in AX-CPT is among the first model-based efforts to understand this tradeoff, which has previously been measured as a behavioral index (the difference between AY and BX performance; e.g. Braver, 2012). This behavioral index, like our model, indicates that subjects in our accuracy-preferring group focus more on context, as indexed by a reduction in BX errors. However, the behavioral index has no way of making the distinction between encoding-oriented and retrieval-oriented contextual effects, a distinction that our model captures. Such multidimensional, quantitative characterization of strategic variability in the processing of perceptual and memory stimuli may pave the way to a richer understanding of context processing both in normal and in diseased populations.

References

Barthelmé, S. and Chopin, N. (2014). Expectation Propagation for Likelihood-Free Inference. *Journal of the American Statistical Association*, 109(505):315–333.

Bitzer, S. and Kiebel, S. J. (2015). The Brain Uses Reliability of Stimulus Information when Making Perceptual Decisions. In *Advances in Neural Information Processing Systems 28*, pages 1–9.

Bogacz, R., Brown, E., Moehlis, J., Holmes, P., and Cohen, J. D. (2006). The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological Review*, 113(4):700–65.

Bogacz, R. and Gurney, K. N. (2007). The basal ganglia and cortex implement optimal decision making between alternative actions. *Neural Computation*, 19(2):442–77.

Braver, T. S. (2012). The variable nature of cognitive control: a dual mechanisms framework. *Trends in Cognitive Sciences*, 16(2):106–13.

Braver, T. S., Barch, D. M., Keys, B. a., Carter, C. S., Cohen, J. D., Kaye, J. A., Janowsky, J. S., Taylor, S. F., Yesavage, J. a., Mumenthaler, M. S., Jagust, W. J., and Reed, B. R. (2001). Context processing in older adults: Evidence for a theory relating cognitive control to neurobiology in healthy aging. *Journal of Experimental Psychology: General*, 130(4):746–763.

Cohen, J. D., Barch, D. M., Carter, C., and Servan-Schreiber, D. (1999). Context-processing deficits in schizophrenia: Converging evidence from three theoretically motivated cognitive tasks. *Journal of Abnormal Psychology*, 108(1):120–133.

Crump, M. J. C., McDonnell, J. V., and Gureckis, T. M. (2013). Evaluating Amazon's Mechanical Turk as a tool for experimental behavioral research. *PLoS ONE*, 8(3):e57410.

Edwards, W. (1965). Optimal Strategies for Seeking information : Models for Statistics , Choice Reaction Times , and Human Information Processing. *Journal of Mathematical Psychology*, 329:312–329.

Einstein, G. O. and McDaniel, M. a. (2005). Prospective Memory. Multiple Retrieval Processes. *Current Directions in Psychological Science*, 14(6):286–290.

Eysenck, M. W., Derakshan, N., Santos, R., and Calvo, M. G. (2007). Anxiety and cognitive performance: Attentional control theory. *Emotion*, 7(2):336–353.

Gold, J. I. and Shadlen, M. N. (2007). The neural basis of decision making. *Annual Review of Neuroscience*, 30:535–74.

Kira, S., Yang, T., and Shadlen, M. N. (2015). A Neural Implementation of Wald's Sequential Probability Ratio Test. *Neuron*, 85(4):861–873.

Laming, D. R. J. (1968). *Information theory of choice-reaction times*. Academic Press, London.

Lositsky, O., Wilson, R. C., Shvartsman, M., and Cohen, J. D. (2015). A Drift Diffusion Model of Proactive and Reactive Control in a Context-Dependent Two-Alternative Forced Choice Task. In *The Multidisciplinary Conference on Reinforcement Learning and Decision Making*, pages 103–107.

McDonnell, J. V., Martin, J. B., Markant, D. B., Coenen, A., Rich, A. S., and Gureckis, T. M. (2015). psiTurk.

Minka, T. P. (2001). Expectation Propagation for Approximate Bayesian Inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pages 362–369, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, 85(2):59–108.

Ratcliff, R., Gomez, P., and McKoon, G. (2004). A diffusion

model account of the lexical decision task. *Psychological Review*, 111(1):159–82.

Ratcliff, R. and McKoon, G. (2008). The Diffusion Decision Model: Theory and Data for Two-Choice Decision Tasks. *Neural Computation*, 20(4):873–922.

Ratcliff, R. and Rouder, J. N. (1998). Modeling Response Times for Two-Choice Decisions. *Psychological Science*, 9(5):347–356.

Servan-Schreiber, D., Bruno, R. M., Carter, C. S., and Cohen, J. D. (1998). Dopamine and the mechanisms of cognition: Part I. A neural network model predicting dopamine effects on selective attention. *Biological Psychiatry*, 43(10):713–722.

Sheppard, J. P., Raposo, D., and Churchland, A. K. (2013). Dynamic weighting of multisensory stimuli shapes decision-making in rats and humans. *Journal of Vision*, 13(6):1–19.

Shvartsman, M., Srivastava, V., and Cohen, J. D. (2015). A Theory of Decision Making Under Dynamic Context. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2485–2493. Curran Associates, Inc.

Stone, M. (1960). Models for choice-reaction time. *Psychometrika*, 25(3):251–260.

Tanner, W. P. J. and Swets, J. A. (1954). A decision-making theory of visual detection. *Psychological Review*, 61(6):401–409.

Turner, B. M., van Maanen, L., and Forstmann, B. U. (2015). Informing cognitive abstractions through neuroimaging: The neural drift diffusion model. *Psychological Review*, 122(2):312–336.

van Maanen, L., van Rijn, H., and Taatgen, N. (2012). RACE/A: An Architectural Account of the Interactions Between Learning, Task Control, and Retrieval Dynamics. *Cognitive Science*, 36(1):62–101.

van Vugt, M. K., Simen, P., Nystrom, L. E., Holmes, P., and Cohen, J. D. (2012). EEG oscillations reveal neural correlates of evidence accumulation. *Frontiers in Neuroscience*, 6(July):106.

Yu, A. J., Dayan, P., and Cohen, J. D. (2009). Dynamics of attentional selection under conflict: toward a rational Bayesian account. *Journal of Experimental Psychology: Human Perception and Performance*, 35(3):700–17.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Learning the dynamics of Prisoner's Dilemma: Lessons from modeling and simulation

Michael Yu (msyu@cmu.edu)
Department of Social and Decision Sciences, 5000 Forbes Ave
Pittsburgh, PA 15213 USA

Cleotilde Gonzalez (coty@cmu.edu)
Department of Social and Decision Sciences, 5000 Forbes Ave
Pittsburgh, PA 15213 USA

Abstract

Learning to deal with social dilemmas can be difficult as outcomes depend not only on a person's decisions but also on other people's decisions and on how past decisions have changed that environment. We investigate how people might learn about social dilemmas by studying how simulated players using a cognitive model known as instance-based learning (IBL) interact with each other and with a set of fixed strategies in the Prisoner's Dilemma (PD). The current simulation study presents systematic variations in the payoff structure and the other player's strategy. Results indicate that the IBL model can reproduce predicted patterns of cooperation based on the payoff structure and that the model is sensitive to the strategies with which it is matched. The simulations offer explanations of how cognitive processes handle social dilemmas and how the environment of social dilemmas can influence this process.

Keywords: Instance-Based Learning; Cognitive Modeling; Prisoner's Dilemma; Cooperation.

Introduction

Instance-based learning (IBL) is a cognitively-inspired, descriptive model of how we make decisions in dynamic environments, i.e., environments that change over time and in which earlier decisions can inform and influence future actions (Gonzalez, Lerch, & Lebiere, 2003). Dynamic decision making often occurs in repeated social dilemmas, when people are asked to decide between actions that benefit themselves at a cost to the group or benefits the group at a cost to themselves. Past responses can influence how members of the group respond in the future, creating a dynamically complex learning environment. The Prisoner's Dilemma (PD) is a commonly studied social dilemma that instantiates this type of situation in a two-person game.

Classical game theory assumes that players understand explicit information about outcomes, reason about the other player's strategy, and solves for the best solution. Behavioral game theory assumes a similar understanding and adds preferences about the other player's actions and outcomes (e.g., Fehr & Schmidt, 1999; Rabin, 1993). Evolutionary game theory and related models of simulation assume no understanding of the game nor consideration of the other player, but rather that players follow pre-determined strategies (Axelrod & Hamilton, 1981; Danielson, 1992; Messick & Liebrand, 1995). The game is resolved by finding strategies that get better outcomes when different combinations of strategies interact in different ways. Approaches that consider learning including reinforcement learning (C. F. Camerer & Ho, 1998; C. Camerer & Ho, 1999) and applying cognitive models to the PD (Gonzalez, Ben-Asher, Martin, & Dutt, 2015; Gonzalez & Ben-Asher, 2014; Lebiere, Wallach, & West, 2000; Stevens, Taatgen, & Cnossen, 2016) ask how learning responds to changes in the decision-making environment.

This paper integrates approaches from these various traditions to develop a different perspective on learning in the PD. Our primary approach stems from cognitive modeling, using an IBL model to study the learning process. This contrasts with the common methods in classical and behavioral game theory in which the PD can be solved before the players interact. Similar to evolutionary game theory, IBL models interact with and respond to the environment (including payoffs and the strategies of other players), prompting questions of how the environment can change this interaction. Contrasted with evolutionary game theory, the focus of IBL is on learning by individual agents – rather than populations – using paradigms that are more similar to those of repeated interaction in classical and behavioral game theory. Under IBL, changes in behavior are a consequence of dynamic learning rather than population dynamics. A more fundamental assumption with the IBL models that contrasts with much of the classical and behavioral economic research is a focus on descriptive models rather than optimization. While IBL models try to make better decisions, the goal of this paper is *not* to find the "best" way to solve the PD but to understand how the environment influences the nature of decision-making in the PD as well as the stability of those decisions.

The present focus on environmental variation contrasts with much of the previous cognitive modeling research, as well. Similar to evolutionary game theory, the current work uses simulations to support studying a broad range of environments. We adopt general IBL models with robust parameters derived from previous research, but do not emphasize creating a specific model to fit a specific set of human data, as is commonly the case with past cognitive modeling research. The emphasis of the current work is *not* to develop the most comprehensive or best-fitting model of human decision-making in the PD, but to develop a clearer

understanding of how the environment interacts with learning dynamics. Thus, our comparisons to human data are on general trends rather than absolute fit, which may require, as in the case of some of the more sophisticated cognitive models, integration of additional features beyond what a basic IBL learning model provides.

In brief, this paper asks how systematic changes in payoffs and partner strategies influence the learning process of an IBL model. We examine how an IBL model interacts with other simulated players in a repeated PD under various payoff conditions.

Repeated Prisoner's Dilemma

In the Prisoner's Dilemma players pick between two options: cooperate or defect. Payoffs are operationalized as follows. If both players cooperate, each receives a payoff of R (Reward). If one cooperates and the other defects, the cooperator receives S (Sucker) and the defector receives T (Temptation). If both defect, each receives P (Punishment). The PD is defined by the following relationship: $T > R > P > S$, and the best response for each player is to defect. However, if both players cooperate, they would each receive more than they would have had both defected. Repeating the PD with the same partner makes the game more complex, as current decisions can influence future outcomes. Over time, players not only learn what outcomes are likely for any given decision, but also influence the behavior of the other player, changing the likelihood of the different outcomes.

Payoff sensitivity. A common finding in empirical studies of the PD is that the player's likelihood of cooperation is related to the values of T, R, P, and S. One of the strongest representations of this was proposed by Rapoport (1967) as the K-index, with a higher K-index predicting higher cooperation. The K-index was defined as: $K = \frac{R-P}{T-S}$. However, rational economic models always predict defection and no correlation between cooperation and the K-index. According to these models, players can reason that defection is best in the last round, since reputation has no effect after the last round; that defection is the best in the second to last round, since reputation has no effect in the last round; and so on in a process known as backward induction.

In contrast, learning models, such as IBL, expect players to explore their options and learn through trial-and-error. Past research predicts that two IBL models acting independently of each other would result in a decrease of cooperation from early to later rounds (Gonzalez et al., 2015), and IBL models have also been found to be sensitive to the payoffs of the PD, consistent with Rapaport's K-index (Gonzalez & Ben-Asher, 2014). In this research we explore these predictions systematically in a wider range of payoff values of the PD and when the IBL model is paired against various known strategies.

Strategic sensitivity. Outcomes depend not only on payoffs, but on the likelihood of each outcome from each decision. These likelihoods depend on the partner's behavior in the game. In contrast to the reasoned approach taken by rational economic theory, where each player attempts to predict what other will do, evolutionary game theorists assume that players follow predetermined strategies (Axelrod & Hamilton, 1981). Strategies that are successful against other strategies are replicated, whereas unsuccessful strategies are removed from the population, leaving the most 'robust' strategies. However, this approach focuses on population characteristics (groups of agents) with 'learning' occurring over generations, rather than on learning of individual members of the population.

Two better-known strategies in evolutionary game theory are tit-for-tat (TFT) and win-stay-lose-shift (WSLS). TFT cooperates in the first round and, in all following rounds, copies the other player's action from the previous round. Thus, it cooperates if the other player cooperated and defects if the other player defected. WSLS also cooperates in the first round. In all subsequent rounds, WSLS will stick with the decision it made in the last round if it received either R or T; or will change its decision if it received either S or P.

In contrast, IBL models focus on how individuals learn – but how IBL models interact with other strategies has not been explored. As IBL models learn over time, they may adjust to other strategies within one game of the repeated PD.

Instance-based Learning

In IBL models decisions are stored in memory as a unique combination of actions and outcomes. Each pair of action and outcome is referred to as an *instance*. When facing a choice, the model estimates a *blended value* for each action being considered. The action with the highest blended value is selected. The blended value, V, for an action, x, at a point in time, t, is: $V_{xt} = \sum_o p_{xot} u_{xo}$ [Eq. 1], where p_{xot} is the retrieval probability of an outcome, o, associated with the action, x; and u_{xo} is the utility associated with the action, x, and outcome of o. In the PD, the actions can be represented as either "cooperate" or "defect," and the outcomes as T, R, P, and S.

The *retrieval probability* of an instance is influenced by the activation of that instance relative to the sum of instances which include the same action. The retrieval probability, p_{xot}, for action, x, and outcome, o, at a specific time, t, is: $p_{xot} = e^{\frac{A_{xot}}{\sigma\sqrt{2}}} / \sum_x e^{\frac{A_{xot}}{\sigma\sqrt{2}}}$ [Eq. 2], where σ is a noise parameter, A_{xot} is the activation of the instance with action, x, outcome, o, and at time, t.

Activation is higher for instances that were more frequent or more recently observed. The *activation*, A_{xot} for an option, x, and outcome, o, at time t_i is: $A_{xot} = \ln \sum (t - T_{xo})^{-d} + \sigma \ln\left(\frac{1-\gamma_{xot}}{\gamma_{xot}}\right)$ [Eq. 3], where d is a decay parameter, σ is the same noise parameter as in Eq. 2, T_{xo} is the set of all times in which the instances with action, x, and outcome, o were observed, and γ_{xot} is a draw from a uniform distribution bounded by 0 and 1 for the current

action, x, at time t. For example, if "cooperate" is more often and more frequently met with the sucker outcome (S) than the reward outcome (R), then the cooperate/sucker instance would be higher in activation and retrieval probability, and the sucker payoff would more strongly influence the blended value of the "cooperate" option than the reward payoff.

The IBL model often takes standard parameter values from ACT-R, the cognitive architecture from which the Activation mechanism comes from (Anderson & Lebiere, 1998): σ is set to 0.25 and d is set to 0.5. As people are not expected to approach decisions with an empty memory, a common approach is to create *prepopulated instances* that represent initial beliefs about the decisions they expect to experience (Lejarraga et al., 2012). The utility associated with these prepopulated instances is typically set to some value higher than the highest possible observable value from the actual decisions to allow for initial exploration and are entered into memory with a time of 0.

The IBL model presented above is a model of an individual, aware of only (Gonzalez et al., 2015) interacts with different PD environments while controlling for broader social considerations including other-regarding preferences and beliefs.

Simulation Overview

In our simulations, we have an IBL model play the repeated PD with either another IBL model or another strategy over the course of 100 rounds. We simulate 400 pairs of players and focus on three measures: individual cooperation rates; alternation rates (switches from cooperate to defect or defect to cooperate); and how models behave as pairs (mutual cooperation, mutual defection, and mixed cases).

The IBL model follows the definition noted above, with σ set to 0.25 and d set to 0.5. Prepopulated instances for cooperate and defect are included with utilities set higher than the temptation payoff to promote exploration. The utility is set arbitrarily at 1.5 times T, i.e., at 15. When matched with other IBL models or other strategies, the IBL model receives information only about its actions and outcomes and no information about the other model's actions or outcomes.

To examine a range of payoffs, we adapt a method used by Moisan, et al. (2015) and inspired by Rapaport and Chammah (1965) and Axelrod (1967), where the payoffs of the PD are normalized with a fixed value for T and S. In the present simulation, T is fixed at 10, S is fixed at 0, and R and P vary between 0 and 10 in intervals of 1 such that $R \geq P$. Our simulations include boundary cases that are not strictly version of the PD: $T = R = 10$, $P = S = 0$, and $R = P$. These boundary cases give a sense as to how the models may behave close to the limits as $R \to T$, $P \to S$, and $P \to R$. This method also provides different payoff structures that have the same K-index, which occurs for any payoff structure in which R and P have the same difference.

To examine strategic sensitivity, we consider two simple, unconditional and two sophisticated, conditional strategies. For simple strategies, we match the IBL model with models that unconditionally cooperate (All-C) or unconditionally defect (All-D). For sophisticated strategies, we match the IBL model with another IBL model, a TFT and WSLS strategy.

Simulation Results

Figure 1 presents the results of simulations in which two IBL models are paired with varying levels of R and P. The panels indicate the cooperation and alternation rates of one of the IBL models from each pair, with the top of each panel representing 100% cooperation/alternation and the bottom representing 0% cooperation/alternation; and the left of each panel representing the 1st round and the right representing the 100th round. Panels that are lightly shaded represent the boundary conditions and are not 'true' PD games.

Across all games, the average cooperation rate starts at 50% (prepopulated instances cause the models to randomly decide to cooperate and defect in the first round), but drift towards increased cooperation or increased defection over time. Behavior does not necessarily change consistently towards cooperation or defection, e.g., the environment in which $R = 9$ and $P = 1$, shows a slight increase in cooperation before settling at a lower rate of cooperation. Final round behavior (at the far right of each panel, discussed more below) includes a variety of cooperation rates across panels. While this behavior may stabilize at a certain cooperation rate, this does not imply that the players have settled on cooperation or defection. While alternation rates also tend to decrease, for many payoff environments, the alternation rates *do not* trend towards 0. End-of-game behaviors include environments where players may switch back and forth between cooperation and defection.

Payoff sensitivity

Cooperation Rate A logistic regression of the final round cooperation relative to the K-index of each simulation indicates that cooperation by the IBL models increases as the K-index increases, B = 4.225, 95% CI [4.097, 4.355]. The coefficient on an ordinary least squares (OLS) regression provides a more intuitive estimate of the effect of K-index on cooperation, B = 0.681, 95% CI [0.664, 0.698], implying a 68.1 percentage point increase from a K-index of 0 to 1 (or a roughly 6.8% increase for each increase of 0.1). These findings are consistent with predictions that cooperation should increase with a higher K-index (Rapoport, 1967). This is visually confirmed in Figure 1, as each diagonal (bottom left to top right) indicates environments with the same K-index. For example, R = 2, P = 1 and R = 9, P = 8 have the same K-index (0.1) and relatively lower cooperation than both R = 5, P = 1 and R = 9, P = 5 (K-index = 0.4).

The K-index appears to account for most but not all of the influence of payoffs on cooperation. A logistic regression of the final round cooperation on the K-index and the R

payoff[1] shows that the effect of the K-index is an order of magnitude larger than that of the R payoff, $B_{K\text{-}index}$ = 4.830, B_R = -0.109; or with OLS, $B_{K\text{-}index}$ = 0.732, B_R = -0.010.

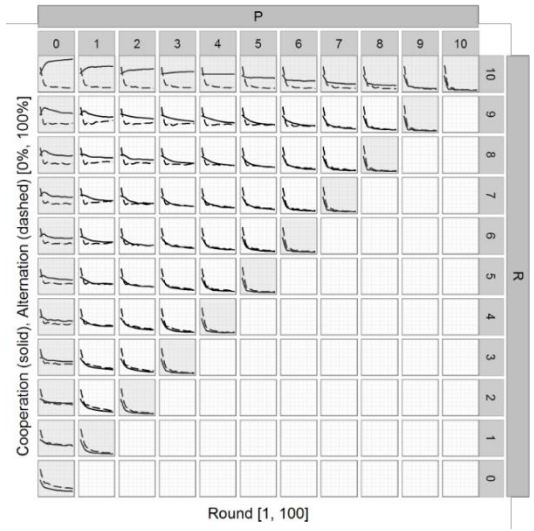


Figure 1: Development of cooperation and alternation for paired IBL models with varying levels of R and P (T = 10, S = 0) across rounds; boundary conditions shaded in gray

Alternation While cooperation rates indicate the general degree of cooperation between players, alternation provides an indication of how stable that cooperation is. Alternation rate may be compared to the concept of evolutionarily stable strategies (ESS) from the evolutionary game theory literature – which suggest how resistant strategies are to changes in the environment (and in particular, invasion by other strategies). In contrast to the idea of ESS, however, higher alternation, while seemingly less stable, may suggest a more robust learning *mechanism*, allowing agents to adapt more quickly from environmental changes.

While distinct from cooperation, alternation is constrained by the cooperation rate. As cooperation rates trend towards 0 or 1, alternation rates trend towards 0. For example, if everyone cooperates in both Round 99 and Round 100, it is impossible for anyone to have switched from defect to cooperate (since all cooperated in Round 99) or switched from cooperate to defect (since all cooperated in Round 100). The highest possible alternation exists where cooperation rates are 50 percent. Nonetheless, cooperation and alternation rates are not perfectly correlated. Environments which produce similar cooperation rates (e.g., R = 10, P = 3, cooperation = 0.293; R = 9, P = 3, cooperation = 0.305) can produce different alternation rates (e.g., R = 10, P = 3, alternation = 0.058; R = 9, P = 3, alternation = 0.210).

Given the non-linear constraints placed by cooperation on the alternation rate, it is unsurprising that alternation is less well predicted by the K-index, although an effect still exists. A logistic regression of alternation between the second-to-last and last rounds on the K-index suggests a positive effect of K-index on alternation, B = 1.400, 95% CI [1.277, 1.525]; or by OLS, B = 0.179, 95% CI [0.164, 0.195]. Including both the K-index and R payoff as parameters in the regressions we find coefficients for the logistic regression of $B_{K\text{-}index}$ = 2.728, B_R = -0.223; and by OLS, $B_{K\text{-}index}$ = 0.285, B_R = -0.021.

Visual inspection of Figure 1 suggests that the pattern of the K-index predicting alternation seems stronger for our 'proper' PDs (non-shaded panels) than for the boundary conditions (shaded panels). As we move towards the upper left panel and higher K-indices, we see that alternation increases. The trend breaks as we approach the boundary condition of $R = 10$ (top row), which shows lower alternation relative to $R = 9$. At this point, the R and T payoffs are identical and there is no temptation motive to draw players from mutual cooperation towards defection. This suggests that the dynamics between temptation and reward may be particularly critical in driving alternation.

Table 1: Pattern of individual and paired behaviors at corner points within simulated Prisoner's Dilemma

K-index	0.8	0.1	0.1
(R, P)	(9, 1)	(9, 8)	(2, 1)
C	44.50	4.25	7.50
Alt	29.50	3.75	13.25
CC	25.50	1.50	2.00
DD	38.25	93.75	87.75
CD/DC	36.25	4.75	10.25

Paired behaviors. Looking at the strategies of *both* players in a pair provides further insight into the social dynamics of the PD. For example, a 50% cooperation rate could be achieved if half of the pairs are engaged in mutual cooperation (CC) and half in mutual defection (DD); or if all pairs include one cooperator and one defector (CD/DC).

Table 1 provides a deeper analysis of the final round behavior for the cases of (R, P) ∈ {(9, 1), (9, 8), (2, 1)}, i.e., the simulations involving the highest/lowest K-indices that are not the boundary conditions. The top half provide results for a single player in each pair, with "C" as the probability of cooperation and "Alt" as the probability of alternation; and the bottom half provides results for the pair of players. The table highlights the differences in the two rightmost payoffs, which have identical K-indices, but with an implicitly painful sucker payoff but little temptation, (9, 8) or a relatively painless sucker payoff but high temptation (2, 1). While there is low cooperation in both cases, the case with high temptation and a low sucker payoff shows more alternation that seems to pull people away from mutual defection in favor of increased mixed pairs ("CD/DC").

[1] Only R or P can be included in addition to the K-index due to multicollinearity.

Strategic sensitivity

Figures 2 and Figure 3 present simulation results of IBL models paired with non-IBL strategies. As with the IBL models paired with other IBL models, the models received no information about the other player's actions or outcomes, but the IBL models outcomes were influenced by the other player's choices. The first set of non-IBL strategies look at unconditional strategies, i.e., strategies that are not influenced by the IBL model's decision; whereas the latter set of non-IBL strategies look at conditional strategies, i.e., strategies which are influenced by the IBL model's decisions either directly (TFT) or indirectly (WSLS). They include panels for different simulations with varying P and R. However, they focus on a subset of P and R (with values of 0, 1, 8, 9 and 10) to allow for a more detailed view of the panels themselves.

All-C/All-D. Figure 2 shows that IBL models paired with strategies that always cooperate or always defect learn to defect quickly. Defection yields the best payoff with no risk of retaliation from these strategies. In contrast, TFT and WSLS would only defect when playing with All-D, but would cooperate with a partner who played All-C.

Exceptions occur only at certain boundary conditions. When paired with All-C, this occurs at $T = R = 10$; that is, when there is no temptation to defect. Cooperating or defecting in such an environment yields the same utility ($T = R$), making indifference reasonable. When paired with All-D, this occurs at $P = S = 0$. Again, cooperating or defecting yields the same utility ($P = S$), and indifference is reasonable. This indifference is reflected in the alternation rate which approaches 50% towards round 100 and is consistent with cooperating and defecting at random.

These findings are consistent with the behavior of individualist human players -- who also defect more against unconditional strategies (Kuhlman & Marshello, 1975), and contrast with strategies, such as TFT or WSLS which would not naturally learn to be opportunistic in these cases.

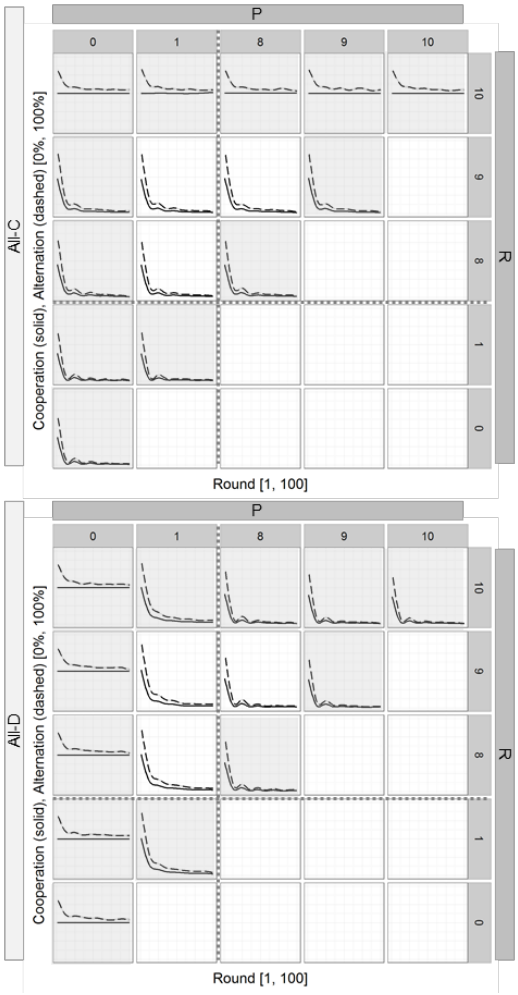


Figure 2: Development of cooperation and alternation for IBL models paired with All-C (top) and All-D (bottom) with select levels of R and P (T = 10, S = 0) across rounds; boundary conditions shaded in gray

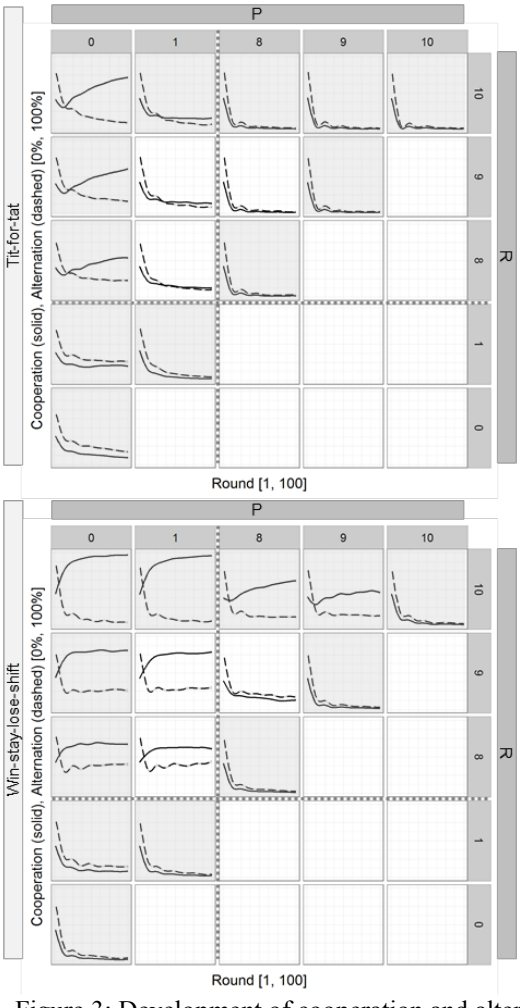


Figure 3: Development of cooperation and alternation for IBL models paired with Tit-for-tat (top) and Win-stay-lose-shift (bottom) with select levels of R and P (T = 10, S = 0) across rounds; boundary conditions shaded in gray

TFT/WSLS Figure 3 highlights the different response of the IBL model when paired with two popular strategies from evolutionary game theory, TFT and WSLS.

Surprisingly, TFT tends towards defection. High defection may be associated with research suggesting that TFT does not do well when their partner behaves inconsistently (Imhof, Fudenberg, & Nowak, 2007), as might be expected when IBL models explore their options. The exception at $P = S = 0$ suggests that the sucker payoff makes cooperation more prohibitive when learning to play with TFT. Not being penalized for moving away from mutual defection, provides an opportunity for the players to arrive at mutual cooperation.

Results for WSLS are similar to those observed in paired IBL models, which may be explained by WSLS having been developed as a simple *learning* model. When paired with a WSLS strategy, cooperation is more greatly affected by reducing the difference in the temptation and reward payoffs (with highest cooperation appearing at the boundary condition of $T = R = 10$) compared to reducing the difference in the punishment and sucker payoffs.

A comparison of Figures 1 and 3 suggest that alternation is higher when an IBL model is paired with WSLS than with other strategies. This is clearer in Figure 4, which shows the relationship between alternation and cooperation in the last round for the different simulations of IBL models partnered with TFT, WSLS, and a second IBL strategy. The relationship shows an upside-down U-shaped curve with all partners, consistent with earlier observations that high and low cooperation rates decrease maximum possible alternation. However, models partnered with a WSLS strategy shows higher alternation at almost all levels of cooperation relative to the TFT and IBL strategies.

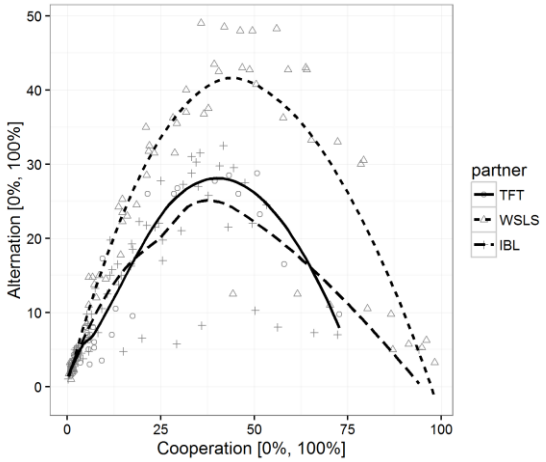


Figure 4: Relationship between cooperation and alternation of the IBL model in final round, when partnered with TFT (circles/solid), WSLS (triangles/short dash), and second IBL (plus/long dash)

Conclusion

Applying simulated cognitive models to social dilemmas helps us to understand how features of a social dilemma specifically impact the learning processes in the absence of other factors, such as other regarding preferences and expectations. The focus on learning differs from classical, behavioral, and evolutionary game theory which do not treat individual learning as a mechanism. The focus on simulation allows us to concentrate on learning more cleanly. For example, previous research finds that cooperation weakly decreases over time when people are paired with a strategy that always cooperates (Lave, 1965; Oskamp, 1971). As this decrease is slight, understanding the nature of this change, or if it is simply noise, can be challenging. By focusing on learning, the results in this paper provides clearer evidence that cooperation may decrease as a result of learning.

Simulations also provide some clearer insight into not only cooperation but the stability of cooperation as highlighted by alternation rates. Our findings suggest that the impact of the strategic environment can influence cooperation and alternation differently. In the case of WSLS, more alternation might draw players out of the 'basin of attraction' represented by mutual defection. Future work might investigate whether high alternation can help players adapt to a changing payoff or strategic environment, given that higher alternation suggests consistent exploration.

The present application of basic cognitive models, such as IBL, to the PD is not intended as a substitute for research using human data or for more complex models that try to fit this data. Indeed, the current research can serve as valuable baseline for such models to better highlight the contribution of specific mechanisms, such as information (Gonzalez et al., 2015), surprise and meta-cognition (C. Camerer & Ho, 1999; Gonzalez & Ben-Asher, 2014; Stevens et al., 2016), and initial beliefs (Lebiere et al., 2000). Altogether, the current research suggests additional areas of investigation and potential boundary conditions under which those models might be tested.

The present work can also be seen as an application of the methods from evolutionary game theory into cognitive modelling, in which we study how varying environments can impact learning rather than population dynamics. Similar to some of that research in that work, we can use simulations across multiple levels of variables to develop a map of sorts in terms of understanding potential areas of investigative interest and guiding our expectations of what results may be likely. The use of basic cognitive models to social dilemmas can help us better understand how we learn and how our approach to games develops over time.

Acknowledgments

This research was supported by the Network Science Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory or the U.S. Government.

References

Axelrod, R. (1967). Conflict of Interest: An Axiomatic Approach. *The Journal of Conflict Resolution*, *11*(1), 87–99.

Axelrod, R., & Hamilton, W. D. (1981). The evolution of cooperation. *Science*, *211*, 1390–1396.

Camerer, C. F., & Ho, T. (1998). Experience-Weighted Attraction Learning in Coordination Games: Probability Rules, Heterogeneity, and Time-Variation. *Journal of Mathematical Psychology*, *42*(2/3), 305–26.

Camerer, C., & Ho, T.-H. (1999). Experience-weighted attraction learning in normal form games. *Econometrica*.

Danielson, P. (1992). *Artificial morality: Virtuous robots for virtual games*. Routledge.

Fehr, E., & Schmidt, K. M. (1999). A theory of fairness, competition, and cooperation. *The Quarterly Journal of Economics*, *114*(3), 817–868.

Gonzalez, C., & Ben-Asher, N. (2014). Learning to cooperate in the Prisoner's Dilemma: Robustness of predictions of an instance-based learning model. In *35th Annual Neeting of the Cognitive Science Society (CogSci2014)* (pp. 2287–2292).

Gonzalez, C., Ben-Asher, N., Martin, J. M., & Dutt, V. (2015). A cognitive model of dynamic cooperation with varied interdependency information. *Cognitive Science*, *39*(3), 457–495.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, *27*(4), 591–635.

Imhof, L. A., Fudenberg, D., & Nowak, M. A. (2007). Tit-for-tat or win-stay, lose-shift? *Journal of Theoretical Biology*, *247*(3), 574–580.

Kuhlman, D. M., & Marshello, A. F. (1975). Individual differences in game motivation as moderators of preprogrammed strategy effects in prisoner's dilemma. *Journal of Personality and Social Psychologyocial Psychology*, *32*(5), 922–931.

Lave, L. B. (1965). Factors affecting Co-operation in the Prisoner's Dilemma. *Behavioral Science*, *10*(1), 26–38.

Lebiere, C., Wallach, D., & West, R. L. (2000). A memory based account of the Prisoner's Dilemma and other 2x2 games. In *Proceedings of International Conference on Cognitive Modeling* (pp. 185–193).

Messick, D. M., & Liebrand, W. B. G. (1995). Individual heuristics and the dynamics of cooperation in large groups. *Psychological Review*, *102*(1), 131–145.

Oskamp, S. (1971). Effects of programmed strategies on cooperation in the Prisoner's Dilemma and other mixed-motive games. *Journal of Conflict Resolution*, *15*(2), 225–259.

Rabin, M. (1993). Incorporating Fairness into Game Theory and Economics. *The American Economic Review*, *83*(5), 1281–1302.

Rapoport, A. (1967). A note on the "index of cooperation" for Prisoner's Dilemma. *Journal of Conflict Resolution*, *11*(1), 100–103.

Rapoport, A., & Chammah, A. M. (1965). *Prisoner's dilemma: A study in conflict and cooperation*. University of Michigan Press.

Stevens, C. A., Taatgen, N. A., & Cnossen, F. (2016). Instance-Based Models of Metacognition in the Prisoner's Dilemma. *Topics in Cognitive Science*, *8*(1), 322–334.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Adversaries Wising Up: Modeling Heterogeneity and Dynamics of Behavior

Yasaman Dehghani Abbasi[1], Noam Ben-Asher[3], Cleotilde Gonzalez[2], Don Morrison[2], Nicole Sintov[1], Milind Tambe[1]

[1]{ydehghan,sintov,tambe}@usc.edu; [2]{coty, dfm2}@cmu.edu; [3]nbenash@us.ibm.com

[1]941 Bloomwalk, SAL 300, University of Southern California, SAL (300), Los Angeles, CA 90089, USA

[2] Social and Decision Sciences, 5000 Forbes Avenue, BP 208, Carnegie Melon University, Pittsburg, PA 15213, USA

[3]US Army Research Labs & IBM T.J.Watson Research, 1101 Route 134 Kitchawan Rd, Yorktown Heights, NY 10598

Abstract

Security is an important concern worldwide. Stackelberg Security Games have been used successfully in a variety of security applications, to optimally schedule limited defense resources by modeling the interaction between attackers and defenders. Prior research has suggested that it is possible to classify adversary behavior into distinct groups of adversaries based on the ways humans explore their decision alternatives. However, despite the widespread use of Stackelberg Security Games, there has been little research on how adversaries adapt to defense strategies over time (i.e., dynamics of behavior). In this paper, we advance this work by showing how adversaries' behavior changes as they learn the defenders' behavior over time. Furthermore, we show how behavioral game theory models can be modified to capture learning dynamics using a Bayesian Updating modeling approach. These models perform similarly to a cognitive model known as Instance-Based-Learning to predict learning patterns.

Keywords: Cognitive Models, Decision Making, Artificial Intelligence, Game Theory

Introduction

Building effective defense strategies requires a profound understanding of adversary goals and behaviors. This can be achieved by constructing models that predict the adversary's attack patterns. For example, to have an optimized patrolling strategy for the defender, it is crucial to understand and model adversary behavior and defender-adversary interactions. Given that defense resources are limited, computational models also provide a method for optimizing resource allocation to maximize defense efficiency using the minimum quantity of resources.

To this end, researchers have used insights from Stackelberg Security Games (SSGs) to offer solutions that optimize defense strategies (Korzyk, Conitzer, & Parr, 2010; Tambe, 2011). Generally, SSGs model the interaction between a defender and an adversary as a leader-follower game (Tambe 2011), in which a defender plays a particular defense strategy (e.g., randomized patrolling of an airport's terminals) and then, having observed the defender's strategy, the adversary takes an action. Traditionally SSG research assumes a perfectly rational model of the adversary's behavior, but recent advances have shown this is not a valid assumption. To overcome the limitations of this assumption, bounded rationality models from behavioral game theory have been adopted in recent SSG work, such as the Quantal Response behavior model (McFadden 1976, Camerer 2003). These models, however, typically assume a homogeneous adversary population, creating a single adversary behavior model (Kar et al., 2015).

Again, this assumption has been challenged, and recently some researchers modeled heterogeneous behavior by either assuming a smooth distribution of the model parameters for the entire adversary population (Yang et al., 2014), or by using a single behavioral model for each adversary (Haskell et al., 2014; Yang et al., 2014). In a recent study, using data collected in an Opportunistic Security Games (OSGs), Abbasi et al. (2016) demonstrated that a population of human attackers can be naturally divided into clusters, according to their exploration of the choice options. Furthermore, exploration is negatively correlated with utility maximization, leading to different attack strategies.

The current paper addresses possible limitations in Abbasi et al. (2016) study. Participants' exploration in their experiment was, to some extent, determined by a random termination rule in the game. That is, the number of decisions that participants could make was, at least in part, determined by chance. This particular effect may have contributed to the variability in exploration processes observed. This paper presents a new experimental study using the same OSG used in Abbasi et al. (2016), but enforced a fixed number of decisions for all participants. Furthermore, the larger number of trials in the present study enables the study of human behavior over-time.

With new experimental data, we demonstrate that the population of human attackers found in Abbassi et al. (2016) is robust to the number of decisions that participants make, and not determined by chance. We replicate the clusters of adversarial behavior. Furthermore, given that all participants had a fixed number of decisions in the game, we are able to investigate the adversary behavior dynamics. We show that the categories of adversarial behavior change over the course of the game, and adversary behavior shifts among these categories as adversaries learn the defender's behavior over time.

To account for the change in adversary behavior, we modified traditional economic models of bounded rationality models used in Abbassi et al. (2016), with a Bayesian update method, so that these models would also be able to predict behavior over time. These models are compared to an Instance-Based Learning (IBL) model that provides a cognitively-plausible account for overtime behavior in the OSG.

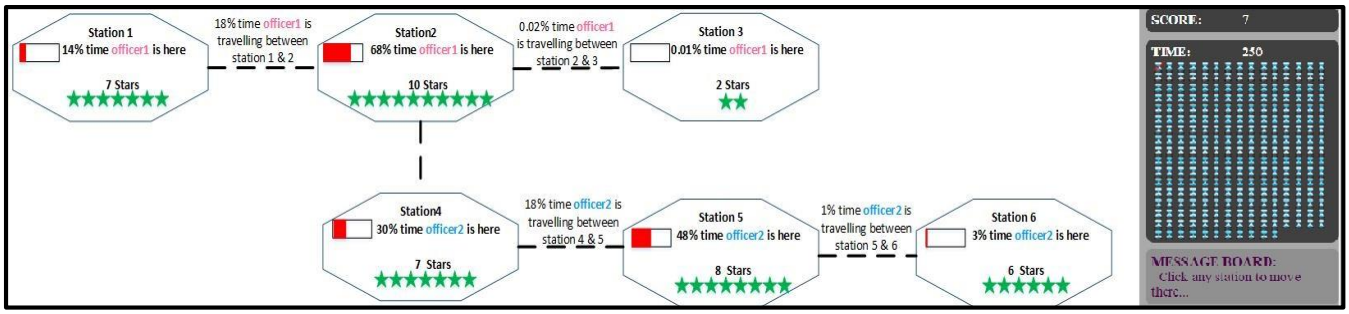


Figure 1: Experiment Game Interface

Experimental Study of Adversarial Behavior in an Opportunistic Security Game

Methods.

Game Design. To collect data on adversarial behavior in the OSG domain, we adopted the experimental design used in Abbasi et al. (2015) using a simulation of urban crime in a metro transportation system with six stations (Figure 1). The players' goal is to maximize their score by collecting rewards (represented by stars in Figure 1) in limited time while avoiding officers on patrol. Each player can visit any station, including the current one. The player can travel to any station by train as represented by the dashed lines in Figure 1. Visiting a station takes one unit of time, and traveling to a new station takes a number of time units equal to the minimum distance between source and destination station along train routes (i.e. for the transportation system represented in Figure 1, if the player is currently at station 1, revisiting station 1 will take one unit of time, and visiting station station 4 will take three units of time). By traveling to a station (or visiting the current station), the player attacks that station and collects the rewards.

Two officers are protecting these six stations. Each officer protects three stations where his patrolling strategy (i.e. probability of officers' presence at each station or route,) is determined by an optimization algorithm similar to the one presented in Zhang et al. (2014). This algorithm utilizes opportunistic adversary behavior model to provide optimized defender strategies.

The stationary coverage probabilities for each station and trains are provided to the players, so players can determine the chance of encountering an officer at a station by considering the percentage of the time that officers spend on average at each station and on a train. However, during the game, the players cannot observe the officers unless they encounter the officer at a station.

The game can finish either if the player uses up all the 250 units of available time in each game, or the game is randomly terminated after the 50th attack with a 10% probability shown by the randomized terminator. The randomized terminator is shown as a fortune wheel with two parts. One part has 10% of the area and colored red, the other part is 90% of the area and colored green. If the arrow stops at the red area, the game is over, and if it lands on the green area, the participant will continue playing the game.

To encourage participants to make each decision responsibly, we showed the randomized terminator to players from the first attack, but for the first 50 trials, we forced the arrow to stop at the green area and start using random number generator after the 50th trial.

In the end, a player's objective is to maximize his total reward in limited time. The player must carefully choose which stations to attack considering the available information about available time, rewards, and officers' coverage distribution on stations and time spent to attack the station. If there is no officer at the station the player has attacked, his score will be increased by the number of stars at the station. If there is an officer at the station, his score remains the same.

Procedures. Each participant began by playing two practice rounds to become familiar with the game. Next, participants played [50+] trials on the main game from which data were used in analyses. We constructed four different graphs (i.e., layouts), each of which had six stations with a different route structure and patrolling strategy. Each participant was randomly assigned to play two practice rounds and the main game on a single graph.

Participants. Participants were recruited from Amazon Mechanical Turk. They were eligible, if they were living in the United States, had previously played more than 500 games and had an acceptance rate of a minimum of 95%.

To motivate the subjects to play games, they were compensated based on their total score ($0.01 for each gained point) in addition to a base compensation ($1). In total, 215 participants took part in the game and went through a validation test (correctly answered all the questions about the game at the end of the instructions). Data from 24 participants who did not pass validation were excluded from further analyses.

Results: Adversarial Attack Patterns

Abbasi et al. (2016) showed that attackers can be divided into distinct groups based on their exploration behavior (i.e., Mobility Score): the ratio of the number of movements between stations over the number of trials (total number of possible movements) by a participant in the game. Therefore, attacking the same station in consecutive trials resulted in a low mobility score while attacking a different station in each trial resulted in a high mobility score. We define three attack patterns: (i) Low Mobility for attackers who did little or no

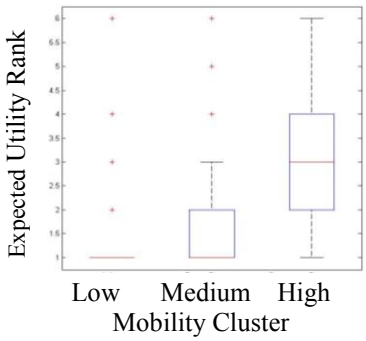


Figure 2: Utility Rank by Cluster

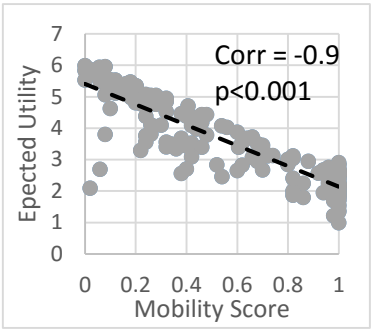


Figure 3: Clustering Distribution

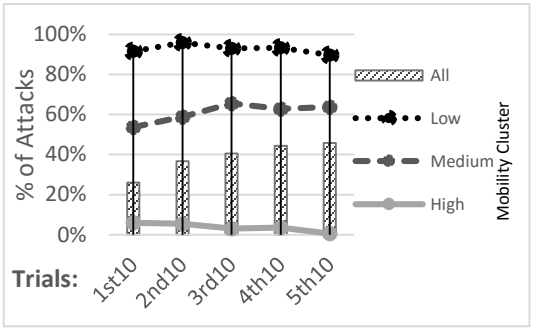


Figure 4: % of attacks on the highest EU station

exploration; (ii) High Mobility for attackers who tended to explore and frequently move between stations and (iii) Medium Mobility for attackers who engaged in a middling level of exploration.

So in this study, in contrast with the previous study, participants had the same number of trials, allowing us to factor out the "variability in the number of decisions" and test whether clusters still emerge from Mobility.

Following Abbasi et al. (2016), we applied hierarchical clustering approach to the data. Participants naturally divided into three groups: participants whose mobility score is less than or equal to 10% or Low Mobility (i.e. Low), participants whose mobility score is greater than or equal to 70%: High Mobility (i.e. High), and participants whose mobility score is greater than 10% and less than 70%: Medium Mobility (i.e. Medium).

Figure 2 shows the three clusters and their corresponding distribution of the Expected Utility Rank of the choices they made (EU-rank distribution). Expected Utility is defined as:

$$EU = \frac{(1 - stationary\ coverage) * reward}{time}$$

Note that the higher expected utility, the better performance. To normalize the utility score among graphs, we have used the ranking of stations' utility instead of its absolute value (the highest utility in the graph is ranked 1).

Participants who belong to the Low Mobility Cluster focused on the stations with highest expected utility (mean=1.2, SD=0.5). On the other hand, participants who tended to move frequently between different stations (High Mobility) attacked average stations with lower utility (mean=3.0, SD=1.3). Participants in Medium Mobility Cluster also attacked a variety of stations but were leaning (on average) towards higher utility rank stations (mean=1.7, SD=1.15). These results replicated those in Abbasi et al., (2016). The robustness of these observations is very important in designing defenders' strategies as they show that attackers that belong to different clusters make decisions differently.

The cluster results are reinforced by Figure 3 that illustrates the negative correlation between mobility scores and the average utility of the attacked stations by the participant (r= -0.9, p<.001). Similarly, there is a significant negative correlation (r = -0.36, p <0.001) between the final score of the participant and his mobility score. In other words, the more participants moved between stations, the lower their score was, and the less money they earned in the experiment. The main question of interest in this research is the change of adversarial behavior over the course of the 50+ trials. We expect that as attackers play the game they will learn to discover the station with higher EU and the patterns of defense behavior, and therefore learn to concentrate in the most profitable stations.

To test this hypothesis we analyzed participants' behavior over the course of the 50 trials. Figure 4 demonstrates the percentage of attacks on the stations with the highest Expected Utility (EU) in each of 5 consecutive blocks of 10 trials each (each participant was re-assigned to different clusters based on his mobility scores in each of the five blocks).

For the participants who belong to Low Mobility and High Mobility Clusters, the percentage of attacks has small fluctuation over time. On the other hand, this percentage increase for the participants in the Medium Mobility Cluster. Moreover, the bar charts show the percentage of attacks on the highest EU stations by all the participants combined. Over the course of the 50 trials, the percentage increases significantly as the percentage of Low Mobility participants increases over time while the percentage of High Mobility participants decreases (Figure 5).

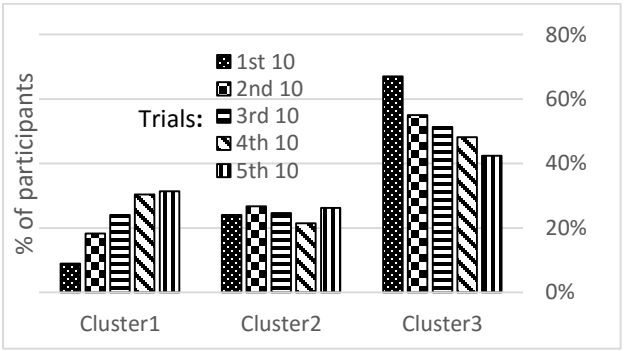


Figure 5: Cluster distribution over time

In other words, participants' shifts toward clusters with lower mobility score and higher rationality level over time. These observations provide us with further insights for designing defenders' strategies, as we show that in addition to classifying attackers by their mobility behavior we also need to consider how adversaries become smarter and learn the defend strategy with more attack attempts.

Models of Adversarial Behavior in OSGs

In the following, we present competing models that represent adversarial behavior and focus on a modification to these models that capture changes in human behavior over time.

Quantal Response Model (QR)

Quantal Response model captures the bounded rationality of a human player through the uncertainty in the decisions making process (McKelvey & Palfrey 1995; McFadden 1976). Instead of maximizing the expected utility, QR posits that the decision maker chooses an action that gives a high expected utility, with a probability higher than another action which gives a lower expected utility. In the context of OSG, given the defender's strategy s (e.g., stationary coverage probability at station i (s_i) shown in Figure 1), the probability of the adversary choosing to attack target i when in target$j$,$q_{i,j}(s)$, is given by the following equation:

$$q_{i,j}(s) = \frac{e^{\lambda * EU_{i,j}(s)}}{\sum_{1 \le k \le 6} e^{\lambda * EU_{(k,j)}(s)}}$$

where λ is his degree of rationality (higher value of λ corresponds to higher rationality level) and $EU_{i,j}(s)$ is the expected utility (EU) of the adversary as given by:

$$EU_{i,j}(s) = \frac{r_i}{time\ (i,j)} * (1 - s_i)$$

Where r_i is the number of stars at station $i, time\ (i,j)$ refers to time taken to attack station i when a player is in station j

Subjective Utility Quantal Response (SUQR)

The SUQR model combines two key notions of decision making: Subjective Expected Utility, SEU, (Fischhoff et al., 1981) and Quantal Response; it essentially replaces the expected utility function in QR with the SEU function (Nguyen et al., 2013). In this model, the probability that the adversary chooses station i when at station j, when the defender's coverage is s, is given by $q_{i,j}(s)$. $SEU_{i,j}$ is a linear combination of three key factors. The key factors are (a) r_i, (b) s_i, and (c) $time_{i,j}$, $w = <w_r, w_{sta}, w_{time}>$ denotes the weights for each decision making feature:

$$q_{i,j}(s) = \frac{e^{SEU_{i,j}}}{\sum_{t' \in T} e^{SEU_{i,t}}} \text{ where}$$

$$SEU_{i,j} = w_r . r_i + w_{sta} . s_i + w_{time} . time_{i,j}$$

Bayesian Update of Human Behavior Models

In the previous study (Abbasi et al., 2016), the human behavior models did not have the power to predict how human behavior changed over time. On the other hand, our results show participants learn to take advantage of the defense algorithm, and they attack stations with higher utility over the course of the trials (becoming wiser). Thus, it is important that models of adversarial behavior account for the change in participants' behavior. Furthermore, in Abbasi et al. (2016) the QR and SUQR models were compared to a process model, a cognitive model that predicts individual choices over time (the IBL model). In some way, these comparisons did not demonstrate the most important advantages of a cognitive model of learning: to predict individual choices at each point in time. For this reason and given our experimental results, we modified the traditional QR and SUQR models described above with a Bayesian update method (B-QR and B-SUQR), so that these models would make predictions more similar to those that the IBL model can make, with individual choices over time.

To use the Bayesian update method, we focus on participants' decision at each trial: each participant made a decision of selecting one out of the six stations to attack. We modeled this problem with a Multinomial distribution over six options with a probability vector $< p_1, \dots, p_k >$ where p_i refers to probability of choosing option i in each trial.

At first, before having any data, we assumed that participants can attack any of the six stations with uniform probability. Then, after each ten trials, we gathered data on the actual number of attacks at each station, yielding data on the actual probability of attacking each station. So $< p_1, \dots, p_k >$ can be updated in the Multinomial Distribution. Luckily, Dirichlet distribution (Bernard, J. M., 2005) is a conjugate distribution for the Multinomial distribution which leads to generating a distribution for each of the probabilities in Multinomial distribution. So in Bayesian-QR and Bayesian-SUQR, after each 10 trials, the distribution over probabilities of attacks get updated and then 100 random samples generated out these probability distributions and 100 human behavior models' parameters were extracted using these samples of probability of attaching each target.

Instance-Based Learning Model

The IBL model (Gonzalez & Dutt, 2011; Lejarraga, Dutt & Gonzalez, 2013) of an adversary makes a choice about the station to go to each trial by using the Blended Value. The Blended value V represents the expected value of attacking each station (option j) in a particular trial:

$$V_j = \sum_{i=1}^{n} p_{ij} x_{ij}$$

where x_{ij} refers to the value (payoff) of each station (the number of stars divided by time taken) stored in memory as instance i for the station j, and p_{ij} is the probability of

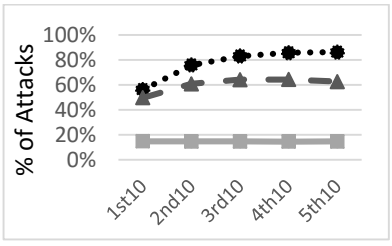


Figure 6: % of attack on the highest EU station predicted by IBL

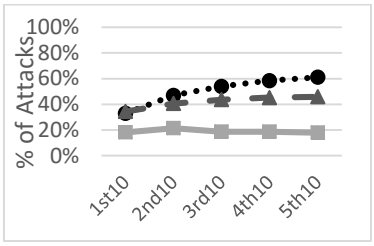


Figure 7: % of attack on the highest EU station predicted by B-QR

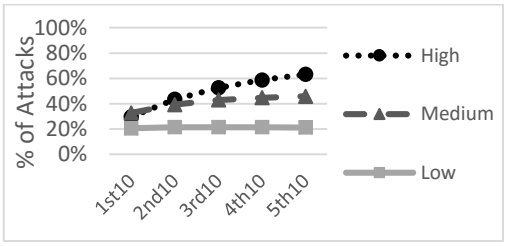


Figure 8: % of attack on the highest EU station predicted by B-SUQR

retrieving that instance for blending from memory (Gonzalez & Dutt, 2011; Lejarraga et al., 2012) defined as:

$$p_{ij} = e^{\frac{A_i}{\tau}} \Big/ \sum_l e^{\frac{A_l}{\tau}}$$

Where l refers to the total number of payoffs observed for station j up to the last trial, and τ is a noise value defined as $\sigma \cdot \sqrt{2}$. The σ variable is a free noise parameter. The activation of instance i represents how readily available the information is in memory:

$$A_i = ln \sum_{\substack{t_p \\ \epsilon\ observed}} (t - t_p)^{-d} + \sum_{\substack{Attribute \\ \epsilon\ Situation}} P(M_{Attribute} - 1) + \sigma \ln\left(\frac{1-\gamma_{i,t}}{\gamma_{i,t}}\right)$$

Please refer to (Anderson & Lebiere, 1998) for a detailed explanation of the different components of this equation. The Activation is higher when instances are observed frequently and more recently. For example, if an unguarded, nearby station with many stars (high reward) is observed many times, the activation of this instance will increase, and the probability of selecting that station in the next round will be higher. However, if this instance is not observed often, the memory of this station will decay with the passage of time (the parameter *d*, the decay, is a non-negative free parameter that defines the rate of forgetting). The noise component σ is a free parameter that reflects noisy memory retrieval.

Importantly, in addition to the kernel mechanisms of the IBL model described above, and used in a multitude of studies (see Gonzalez, 2013 for a summary), Abbasi et al. (2016) proposed a mechanism that would allow the IBL model to account for the various mobility clusters. This mechanism was a randomization rule applied at each time step, which resulted in making a random selection of a station instead of selecting the station with the highest *Blended* value. This randomization rule served the purpose of generating the clusters of participants with diverse mobility scores. In the current work, this rule was removed given that each participant made exactly 50 choices, and the process of learning over those should be captured by the kernel mechanisms of the IBL model without the additional randomization rule.

Modeling Results

To test the models, we divided the human data set into two groups: training and test datasets. For each cluster or block of trials, 70% of the participants were randomly selected, and their data were used to train the QR, SUQR, and their Bayesian versions (B-QR and B-SUQR) and to fit the *d* and the σ in the IBL model. The remaining 30% of the participants were used for testing the models.

For comparison of different models, we use Root Mean Squared Error (RMSE) representing the deviation between a model's predicted probability of an adversary's attack ($\hat{p}$) and the actual proportion of attacks from each station to others in the human data (p).

$RMSE(\hat{p}) = \sqrt{MSE}(\hat{p})$ where MSE $(\hat{p}) = \frac{1}{n}\sum(\hat{p} - p)^2$

Table 1 shows the results on the full data set. Although models provide different perspectives, their prediction errors are similar. The IBL model captures learning and decision dynamics over time while QR and SUQR predict the stable state transition probabilities of the attacker while B-QR and B-SUQR[1] update the transition probabilities of attacker after each ten trials. Table 2 shows the performance of different models in different clusters.

Table 1: Metrics and Parameter on the full data set

Model	*Parameters*	*RMSE*
QR	0.41	0.25
SUQR	<2.9,-2.1,-2.7>[2]	0.24
Bayesian-QR (B-QR)	0.34	0.24
Bayesian-SUQR (B-SUQR)	<2.5.-1.9,-2.1>	0.23
IBL	<0.01,0.01>[3]	0.23

In Low Mobility Cluster, human behavior models outperform IBL model, and the Bayesian update on these models results in a significant improvement over their counterparts models. For the Medium Mobility Cluster the improvement is not significant, and all models' prediction errors are similar for the High Mobility Cluster.

[1] For Bayesian-QR (B-QR) and Bayesian-SUQR (B-SUQR), the average values over 100 data have reported in the tables

[2] $<w_r, w_{sta}, w_{time}>$

[3] $<noise, decay>$

Table 2: Metrics and Parameters on each Cluster

Clusters	*Model*	*Parameters*	*RMSE*
Low Mobility Cluster	***QR***	1.28	0. 33
	SUQR	<5.6,-4.4,-8.9>[2]	0.35
	B-QR	0.69	0.20
	B-SUQR	<2.8,-2.2,-5.1>	0.17
	IBL	<0.46, 0.01>[3]	0.50
Medium Mobility Cluster	***QR***	0.69	0.27
	SUQR	<4.5,-2.3,-5.1>[2]	0.28
	B-QR	0.49	0.17
	B-SUQR	<3.0,-1.7,-2.5>	0.24
	IBL	<3.64, 1.82>[3]	0.35
High Mobility Cluster	***QR***	0.07	0.25
	SUQR	<2.1,-1.7,-0.5>[2]	0.25
	B-QR	0.14	0.27
	B-SUQR	<1.9,-1.5,-1.5>	0.28
	IBL	<0.1, 2.71>[3]	0.27

For the Bayesian models, the reported parameters in the tables were averaged over extracted parameters from the samples. Further analyses over these parameters are shown in Figure 9 which shows the distribution of Quantal Response λ-value over time. As shown in the graph, the λ-value increases, which means the participants are becoming more rational.

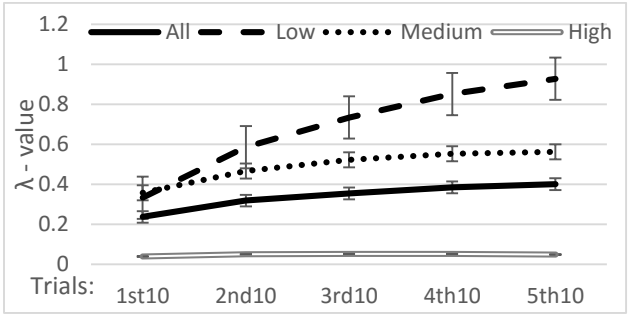


Figure 9: lambda value over time

This observation is consistent with Figure 4, extracted from participants' data, where the bar chart shows the percentage of attacks on the highest expected utility stations which increase over time.

Figure 6, Figure 7 and Figure 8 all focus on the percentage of attacks on the highest EU stations over time, predicted by the IBL, B-QR, and B-SUQR models, respectively. As shown in the graphs, all models also predict the increasing rationality of the participants over time specifically for Low Mobility Cluster and Medium Mobility Cluster.

Conclusions

In security game researches, understanding human adversary behavior has led to several deployed real-world applications (Tambe 2011), for example, PROTECT for the protection of major ports in the US by the US Coast Guard (Shieh et al. 2012). Although there are a significant amount of such researchers, there has been little research of heterogeneous adversary and how adversaries adapt to defense strategies over time. In this paper, we focus on opportunistic adversaries and advance the prior research which suggested classifying adversary into distinct groups based on the ways humans explore their choice options. More specifically, we advance this work by showing how adversaries shift among the categories as they learn the defenders' behavior over time. Furthermore, we show how behavioral game theory models can be modified to capture the learning dynamics using a Bayesian Updating modeling approach. These models perform similarly to a process model, a cognitive model known as Instance-Based Learning, to predict learning patterns. This study provides interesting insights into building defense strategies. For example, current sophisticated defense algorithms often assume a homogeneous adversary population who behave the same over time. Given the significant impact of modeling adversarial behavior to designing optimum patrolling strategies for the defenders, it is critical to account for this heterogeneity in behavior also we need to have defenders' strategy which adapts to change in human behavior over time.

Acknowledgments

This research was partly supported by the Army Research Laboratory under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA) to Cleotilde Gonzalez. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. This research is also supported by MURI grant W911NF-11-1-0332, and award no. 004525-00001 by US-Naval Research Laboratory.

References

Abbasi, Y. D., Short, M., Sinha, A., Sintov, N., Zhang, Ch., Tambe, M. (2015). Human Adversaries in Opportunistic Crime Security Games: Evaluating Competing Bounded Rationality Models. *Advances in Cognitive Systems.*

Abbasi, Y., Ben-Asher, N., Gonzalez, C., Kar, D., Morrison, D., Sintov, N., Tambe, M. (2016). Know Your Adversary: Insights for a Better Adversarial Behavioral Model. *Proceeding of the Conference of Cognitive Science Society*.

Anderson, J. R., & Lebiere, C. (1998). The atomic components of thought. Lawrence Erlbaum Associates. *Mathway, NJ.*

Bernard, J. M. (2005). An introduction to the imprecise Dirichlet model for multinomial data. *International Journal of Approximate Reasoning*, 39(2), 123-150.

Camerer, C.F. (2003) *Behavioral game theory, Experiments in strategic interaction*. Princeton University Press

Fischhoff, B., Goitein, B., & Shapira, Z. (1981). Subjective expected utility: A model of decision-making. *Journal of*

the American Society for Information Science, 32(5), 391-399.

Gonzalez, C., & Dutt, V. (2011). Instance-based learning: Integrating sampling and repeated decisions from experience. *Psychological review*, 118(4), 523.

Gonzalez, C., Ben-Asher, N., Martin, J. & Dutt, V. (2015). A cognitive model of dynamic cooperation with varied interdependency information. *Cognitive Science,* 39(3), 457-495.

Gonzalez, C., Ben-Asher, N., Oltramari, A., & Lebiere, C. (2015). Cognition and Technology. *Cyber defense and situational awareness*.

Gonzalez, C., Lerch, F. J., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science,* 27(4), 591-635.

Haskell, W., Kar, D., Fang, F., Tambe, M., Cheung, S., & Denicola, L. E. (2014). Robust protection of fisheries with compass. *IAAI* (pp. 2978-2983).

Kar, D., Fang, F., Delle Fave, F., Sintov, N., & Tambe, M. (2015). A game of thrones: when human behavior models compete in repeated Stackelberg security games. *In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (pp. 1381-1390). International Foundation for Autonomous Agents and Multiagent Systems.

Lejarraga, T., Dutt, V., & Gonzalez, C. (2012). Instance-based learning: A general model of repeated binary choice. *Journal of Behavioral Decision Making*, 25(2), 143-153.

McFadden, D. L. (1976). Quantal choice analysis: A survey. In *Annals of Economic and Social Measurement, Volume 5, number 4* (pp. 363-390). NBER.

McKelvey, R. D., & Palfrey, T. R. (1995). Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1), 6-38.

Nguyen, T.M., Yang R., Azaria A., Kraus S., Tambe M. (2013). Analyzing the Effectiveness of Adversary Modeling in Security Games, In *AAAI*.

Shieh, E. A., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., ... & Meyer, G. (2012). PROTECT: An Application of Computational Game Theory for the Security of the Ports of the United States. In *AAAI*.

Tambe, M. (2011). Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned. *Cambridge University Press*.

Yang, R., Ford, B., Tambe, M., & Lemieux, A. (2014). Adaptive resource allocation for wildlife protection against illegal poachers. *In Proceedings of the 2014 International Conference on Autonomous agents and multi-agent systems* (pp. 453-460). International Foundation for Autonomous Agents and Multiagent Systems.

Zhang, C., Jiang, A. X., Short, M. B., Brantingham, P. J., & Tambe, M. (2014). Defending against opportunistic criminals: New game-theoretic frameworks and algorithms. *In Decision and Game Theory for Security* (pp. 3-22). Springer International Publishing.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Toward Integrating Cognitive Linguistics and Cognitive Language Processing

Peter Lindes (plindes@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121 USA

John E. Laird (laird@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121 USA

Abstract

We present a system to comprehend natural language that combines cognitive linguistics with known properties of human language processing. It is built on Embodied Construction Grammar (ECG) and the Soar cognitive architecture. Its core is a novel grounded semantic parser. Experiments show the system produces actionable meanings and fulfills ten cognitive criteria we set out.

Keywords: language comprehension; construction grammar; Soar; grounded semantics; language in robots; cognitive linguistics; cognitive architecture.

Introduction

This work attempts to combine two separate threads of research. One is cognitive linguistics, where formalisms have been developed for syntactic and semantic knowledge, such as Embodied Construction Grammar (ECG; Bergen & Chang, 2013). The second is from research on the cognitive modeling of language processing, where the emphasis is on modeling how humans process language, independent of specific linguistic formalisms for representing syntactic and semantic knowledge. In this paper, we develop a system called LUCIA that attempts to tie these two threads together, developing a novel comprehension system whose knowledge of language is specified in the ECG formalism (Bryant, 2008) and then translated into production rules. Those rules are used in a language comprehension process which is designed to fit many of the characteristics of human language processing.

Cognitive Linguistics

Cognitive linguistics is based on the idea that language is an integral part of cognition. Language is closely related to perception (Miller & Johnson-Laird, 1976) and action (Coello & Bartolo, 2013). To explain language we must study categories (Lakoff, 1987), image schemas (Johnson, 1987; Mandler & Pagán Cánovas, 2014), and metaphor (Lakoff & Johnson, 1980). Meaning is seen as being represented by frames (Fillmore, 1976, 2013; Fillmore & Baker, 2009) or scripts (Schank, 1972). Psychological theories attempt to explain comprehension at the discourse (Kintsch, 1998) and sentence level (Ferstl, 1994). Looking at language usage leads to theories of construction grammar (Goldberg, 1995 & 2006; Hoffmann & Trousdale, 2013) that integrate semantics and syntax.

Construction grammars provide a theory for representing syntax and semantics (Goldberg, 2013). ECG (Dodge, 2010; Feldman, 2006) is a specific formalism in this field based on much of the cognitive linguistic research mentioned above. Such a representation is necessary to language understanding, independent of how the processing is done, in order to insure that the language understanding system is capable of addressing the scope of human language. Parsers have been built for ECG (Bryant, 2008), as well as for a related formalism called Fluid Construction Grammar (FCG), which has been used for communication with robots (Steels & Hild 2012; Steels, 2013). Lindes (2014) used ideas from ECG for information extraction. However, none of these approaches attempts to model the characteristics of human sentence processing.

Consider the ECG example in Figure 1. On the left we see a syntactic construction for a `TransitiveCommand`, and on the right we see a meaning schema called `ActOnIt`, along with its generalization `Action`.

```
construction TransitiveCommand
   subcase of Imperative
   constructional
      constituents
         verb: ActionVerb
         object: RefExpr
   meaning: ActOnIt
      constraints
         self.m.action <--> verb.m
         self.m.object <--> object.m
```

```
schema Action
   roles
      action
      location

schema ActOnIt
   subcase of Action
   roles
      object
```

Figure 1: ECG example

This example shows several characteristics of ECG. A composite construction lists its constituents, in this case named `verb` and `object`. Each constituent slot is labeled with the type of construction that can fill that slot. A construction can specify the name of a meaning schema to be evoked when it is instantiated, in this case `ActOnIt`. Schemas have roles to be filled. Both constructions and schemas can be generalized through the `subcase of` clause, and schemas can inherit roles from their parents. A construction can specify constraints that supply values to these roles through unification. In the example the constraints unify the meanings of the constituents with the roles in this construction's meaning schema.

This formalism is an abstraction that can describe many linguistic structures; however, one unanswered question is:

is this type of representation sufficient for representing the knowledge needed for modeling human sentence processing?

Cognitive Language Processing

Cognitive language processing research (Newell, 1990; Lewis, 1993; Lewis & Vasishth, 2005) looks at building computer models that comprehend language using methods that approximate properties of human language processing. We have chosen to focus on the following characteristics of human-like processing:

1. Incremental – Processing extracts as much syntactic and semantic information as it can from each word, one at a time (Lewis, 1993).
2. Integrated – Syntactic and semantic information are extracted jointly during comprehension (Lewis, 1993).
3. Eclectic – Semantic, pragmatic, and world knowledge are used to resolve ambiguities.
4. Real time – Comprehension proceeds in real time (Lewis, 1993).
5. Useful – The meanings extracted are "actionable intelligence" that the agent can use for its purposes.
6. Repair-based processing – The system greedily builds structures that may need to be repaired as more information becomes available (Lewis, 1993).
7. Context-dependent meaning – Words can have multiple meanings; the meaning in a particular sentence is selected according to the context.
8. Compositional – Elements with known meanings are combined to comprehend novel sentences.
9. Hierarchical – Both lexical items and higher-level constructions contribute elements of meaning (Goldberg, 1995, 2006).
10. Grounded – The meanings derived from a sentence are grounded in the agent's perception, action capabilities, and world knowledge.

Lewis (1993) describes a parser that is incremental (Item 1), does local repairs (Item 6), and shows correspondence to human processing in terms of its real-time performance (Item 4) and the kinds of structures that it has difficulty processing. Lewis and Vasishth (2005) extend this work to explore more detailed mechanisms of memory retrieval. That work, however, does not build full, grounded semantic structures that would be useful to an embodied agent.

Ball et al. (2010), as part of the Synthetic Teammate Project, have a model of human language processing implemented in ACT-R that attempts "adherence to well-established cognitive constraints." This model takes advantage of ACT-R's subsymbolic capabilities to resolve some kinds of ambiguities, and it does incremental, integrated, and grounded sentence understanding (Items 1, 2, and 10). However, the "Double R" theory of grammar it uses does not have the same capabilities of ECG (Feldman et al., 2009) to recognize many alternative expressions and to represent complex semantic structure.

Cantrell et al. (2010) have a system for natural language understanding for robots that is designed to build semantics in an incremental and integrated way (Items 1 and 2), and ground the language in the robot's perception (Item 10). This system, however, does not take advantage of cognitive linguistics or prior work on cognitive language processing.

Bringing these two research threads together has some advantages. Cognitive linguistic theory, and ECG in particular, provide a formal way of describing meaning representations that is grounded in research on human knowledge representation. The formalism also describes syntax and the relationships by which form evokes meaning. Cognitive language processing attempts to ground this theory in actual processing that reflects known characteristics of human processing, thus making a theory that can be tested in the real world.

This brings us to our main research question: is it possible to implement a comprehension system that uses the ECG formalism, that is consistent with human language processing, and that produces results that are useful to an embodied autonomous agent? Here we take some initial steps to answer this question by developing a system based on ECG that has many of the characteristic of human sentence processing.

An Integrated Solution

In this paper we describe LUCIA, which works as part of an embodied Soar agent called Rosie (Mohan et al., 2013). We show that it produces useful results for directing and instructing this robot, and that the method meets the above cognitive characteristics. It does not address the immense scope of natural language, discourse level understanding, the ability to learn new lexical, syntactic, and semantic structure, or how the brain implements comprehension. Nor have we explored the limits of understandable syntactic structures that Lewis (1993) emphasizes.

We have developed a translator that converts ECG into Soar production rules, and we have written by hand a collection of rules that provide the infrastructure for language comprehension. The ECG grammar for our experiments is adequate to comprehend a set of sentences that provide directions to a robot, and the results are evaluated against a gold standard of meaning structures known to be useful to the robot. The outputs from LUCIA produce the correct actions with the Rosie simulator.

In the rest of this paper we explain how LUCIA works, show experimental results of its performance, and discuss how it satisfies the ten properties of human language processing. Then we draw conclusions and propose future work.

Language Processing in LUCIA

Here we describe the basic principles that LUCIA is built on, show some examples, and relate these to our ten items.

Basic Operation

The LUCIA comprehension subsystem replaces the language comprehension part of Rosie and sends messages to the task performance subsystem, which acts on them, as

shown in Figure 2. The comprehension subsystem consists of rules in Soar's procedural memory, some generated from a grammar and some hand-coded. The hand-coded rules encode functionality that is independent of specific language structures.

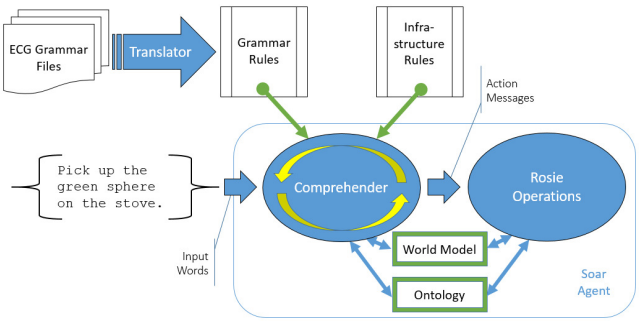


Figure 2: LUCIA in context

Words of a sentence come into the comprehender, which processes them one at a time to create a semantic interpretation of the complete sentence. In doing this, LUCIA draws on a world model that is assembled from the agent's visual perception and an ontology that defines objects, properties, actions, etc. In Soar the rules are held in production memory, the world model in working memory, and the ontology in semantic memory.

When a complete interpretation of a sentence has been built, a message is passed to the task performance subsystem, labeled "Rosie Operations" in Figure 2, which performs the indicated action. This may involve moving the robot, manipulating physical objects, or providing natural language responses to the human user. As the robot acts, it updates its world model, which is always available to the language comprehender.

Linguistic Knowledge

As shown in Figure 1, an ECG grammar consists of "schemas" defining semantic structures and "constructions" which relate an input form to a meaning expressed in those schemas. Our translator is built based on Bryant's (2008) formal definition of the ECG language. Each construction or schema produces one or more Soar rules. In order to have a system that could later be extended to learn more grammar incrementally, each construction or schema is translated independently, without using global knowledge of the grammar or interaction with other items.

The linguistic knowledge that the comprehender depends on is represented in Soar production rules: those generated by the ECG translator, as well as a smaller set of hand-coded rules that provide functions that are common over the whole grammar. These functions include retrieving properties or actions from semantic memory and resolving referential expressions to references to particular objects in the model of the perceived world in working memory. Still others handle bookkeeping tasks.

Dynamic Processing

The core of the system is the `comprehend-word` operator, which is applied once for each input word to implement incremental processing (Item 1). As part of `comprehend-word`, a `lexical-access` operator is selected for each word, and rules generated from ECG apply to create a lexical construction along with any evoked semantic structures. A `match-construction` operator is selected each time one or more constituents can be composed into a larger construction. These operators are applied by other ECG-generated rules which fire, sometimes several in parallel, to evoke, build, and populate semantic schemas. Together, all these rules implement integrated syntactic and semantic comprehension (Item 2). Both lexical and composite constructions contribute meaning (Item 9).

At appropriate points, various hand-coded operators are selected to ground referring expressions to the current perceived world model and the ontology in semantic memory (Item 10). Finally, results for this word are returned to the higher-level state. Once a complete sentence has been comprehended, infrastructure rules interpret it to form a message for the task performance subsystem. These results are compared to the gold standard developed for the robot, so we can verify that they are correct and useful (Item 5).

These operators and rules do not fire in a fixed sequence, but in a dynamic one determined by the word being comprehended, the syntactic and semantic context, and the knowledge contained in the world model and ontology. These dynamics arise from the principle of doing as much analysis as possible while processing each word in order, without any look-ahead to future words (Item 1). This approach can produce good performance, but it often makes mistakes. These are corrected by a local repair mechanism (Item 6) modeled after the one Lewis (1993) used to simulate human sentence processing with Soar.

We call the complete process Informed Dynamic Analysis (IDA) since the syntactic and semantic analyses evolve dynamically together by applying whatever linguistic and world knowledge is relevant at each moment (Item 3).

Examples

Below are examples that illustrate this dynamic process.

Example 1: A Simple Sentence

A simple example is *Pick up the green sphere*. Figure 3 shows the results of the analysis, part of which constitutes an instantiation of the ECG items in Figure 1.

The figure summarizes the operation of the many operators needed to comprehend this sentence. Numbers indicate when structures were built by the corresponding application of `comprehend-word`. Constructions are shown as blue rectangles, their meaning schemas as green ovals, the identifiers of structures in semantic memory in red, and structures in the world model in orange. The identifiers in green and orange are used to make associations between the comprehension process and items in the shared memories.

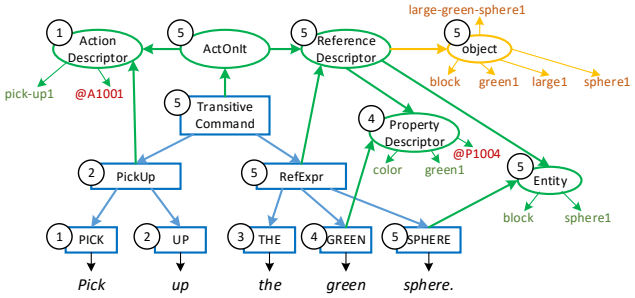


Figure 3: Comprehension of a simple sentence

The semantic parse shown here is built up incrementally as each word is processed in stages 1 to 5 (Items 1 and 2). Each word leads to the retrieval of a lexical construction, those with names in capitals. Larger constructions are composed whenever possible (Items 8 and 9). As soon as the verb is identified in stage 1, its grounded meaning with id `@A1001` is retrieved from semantic memory (Item 10). The `PickUp` construction in stage 2 attaches to the meaning already built for its constituent `PICK`. (The green arrow from `PICK` to its meaning has been omitted to avoid clutter.) In stage 4, a lookup to semantic memory (Item 10) finds the id `@P1004` to ground the property *green*. When the referential expression is complete in stage 5, it is resolved to an object in the world model (Item 10). In stage 5, the complete `TransitiveCommand` construction, a composite of the structures for *Pick up* and *the green sphere*, is also built as soon as its constituents are present. Note that several levels of processing are done for one word (Item 1). No repairs are needed in this example.

Example 2: Phrase Attachment and Repair

Figure 4 shows the abbreviated results for *Pick up the green sphere on the stove*. This example illustrates the integration of Lewis's repair mechanisms with the semantics available from ECG.

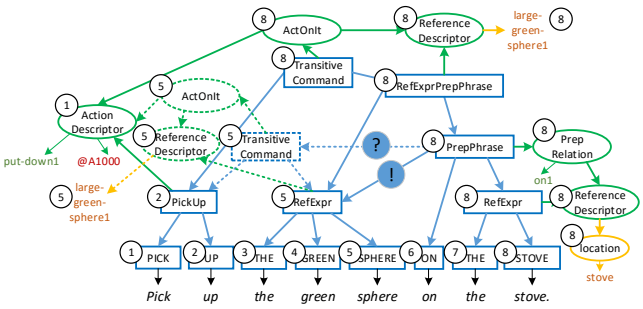


Figure 4: Phrase attachment and repair

In this case, the words up through *sphere* form a valid sentence, so the first 5 stages run exactly as before. But the end has not yet been reached, as *on the stove* remains to be processed. Stages 6 and 7 are very simple, but a lot happens at stage 8. First the process recognizes and resolves *the stove*. Next *on* is added to form a prepositional phrase. Now there is the classic problem of prepositional phrase attachment: should the phrase be attached to the command that is the current upper-most construction, or to modify *the green sphere*?

The simplest way to attach this phrase would be as a target location for the command, and that is what would happen if the sentence were ***Put*** *the green sphere on the stove.* But the system can use semantic knowledge to know that *put* needs a target location and *pick up* does not (Item 3). A "repair" is done by "snipping" (Lewis, 1993) the items shown with dotted lines and attaching *on the stove* to *the green sphere* (Item 6). Now the reference for *the green sphere* must be resolved again with the new information, but in this case the same answer results because in the current perceptual model this sphere is in fact on the stove. Finally, the semantic structure for the command is rebuilt with the revised referential expression.

Attaching a relative clause, as in *Pick up the green block that is on the stove.,* works in a very similar way, except that the word *that* is lexically ambiguous. In this sentence, it is a relative pronoun introducing the relative clause. In *Put that in the pantry.* it is a deictic pronoun referring to something salient in the context. The grammar has both meanings and they both are created during `lexical-access`. Later infrastructure rules select which one to use, and the other is discarded. This illustrates Item 7.

Informed Dynamic Analysis

The whole process just described is similar to the analysis in any semantic parsing system in that it takes a sentence of text and produces a semantic representation. However, it uses a dynamic process where at every step semantic and world knowledge can be applied. Thus, instead of generating many parses and ranking their likelihood, it uses non-syntactic knowledge to resolve ambiguities and repair mistakes dynamically as the analysis proceeds. This approach implements Items 1, 2, 3, and 10.

Experiments

The Rosie team has built up a corpus of several hundred sentences used to instruct the Rosie agent in various tasks. A parser has been custom-built that allows the agent to understand this corpus. The LUCIA system attempts to duplicate the processing of that parser while being more general and scalable to a wider variety of linguistic forms and problem domains. To evaluate the capability, generality, and scalability of LUCIA, we have devised the following experiments.

Experiment 1

First, we took the entire Rosie sentence corpus and reduced it by removing sentences for its game-playing domain, which is beyond the scope of this project, and eliminating duplicate sentences. Then we selected 50 of the remaining 209 sentences. Each of the 50 shows a slightly unique linguistic pattern and they collectively cover much of the linguistic space of all 209 sentences. These 50 sentences fall

into several categories which are listed below, with some of the language forms covered and an example sentence or two for each category:

Declarative statements (8): noun phrases, adjectives, properties, states, prepositional phrases
The red triangle is on the stove.

Manipulation commands (19): manipulation verbs, transitive commands, commands with a location target, prepositional phrase attachment issues, multi-word prepositions
Put the green sphere in front of the pantry.
Store the large green sphere on the red triangle.

Relative clauses, etc. (5): relative clauses with properties, relative clauses with prepositional phrases, multiple prepositional phrases
Pick [up] a green block that is larger than the green box.
Move the green rectangle to the left of the large green rectangle to the pantry.
These two examples show a relative clause, a *larger than* relation that is computed during resolution, a *to the left of* relation which is found stored in the world model and picks out the correct *green rectangle*, and the proper attachment of the two prepositional phrases with *to*.

Navigation commands (10): navigation verbs, spatial references, absolute and relative directions, abbreviated commands, goal phrases
Follow the right wall.
Go until there is a doorway.

Yes/no answers (1): *Yes.*

Definitions of words (2):
Octagon is a shape.

Conditional commands (1):
If the green box is large then go forward.

Questions (4):
What is inside the pantry?
Is the small orange triangle behind the green sphere?

Together, this set of 50 sentences partially addresses the ten distinguishing properties of human sentence processing listed earlier. To cover this set, it was necessary to build the ECG constructions and schemas they use, both for the lexical items and the composite constructions. Then these sentences served as the test suite to fully develop the infra-structure rules that complete the LUCIA comprehender.

We also built an evaluator that takes the output of LUCIA for each sentence and compares it with the gold standard semantics provided by the Rosie team. When differences were found, the grammar and hand-coded rules were corrected as needed to get the desired result. Finally, all 50 sentences were comprehended correctly.

Table 1 shows the number of Soar rules that were generated automatically and by hand. The ECG column counts constructions and schemas, and the Rules column counts Soar production rules. Over 60% of the code was generated automatically from the grammar, showing that the ECG representation is capable of representing the majority of the knowledge that is needed.

Table 1: Experiment 1 statistics

Category	ECG	Rules	Proportion
Grammar	226	487	62.5%
Hand-coded	0	292	37.5%
Total	226	779	

Another key measure of performance relates to real time, our Item 4. The Soar theory (Newell, 1990) maps execution time to real time by assuming each decision cycle takes 50 msec. Lewis (1993, p. 13) points out that humans comprehend speech "as quickly as we hear it" and read even faster at "~240 words per minute." Thus an incremental comprehender has about 4 to 5 decision cycles, on average, to comprehend each word.

Our run of all 50 sentences processed 284 words in 2,582 decision cycles, or 9.09 cycles/word and 132 words/minute. This is too slow by about a factor of two. However, an analysis shows that within a sentence there are 4 decision cycles of overhead within each `comprehend-word` cycle, and this overhead could be reduced considerably.

As we developed the system to comprehend more and more of the 50 sentences, new declarative knowledge in the form of ECG items and new procedural knowledge in the form of the hand-coded rules were added to the system in many small steps. Although LUCIA has no built-in learning mechanism, this increase of knowledge can be thought of as a model of what a true learning system would have to learn. Figure 5 shows how this knowledge grows with the number of sentences comprehended.

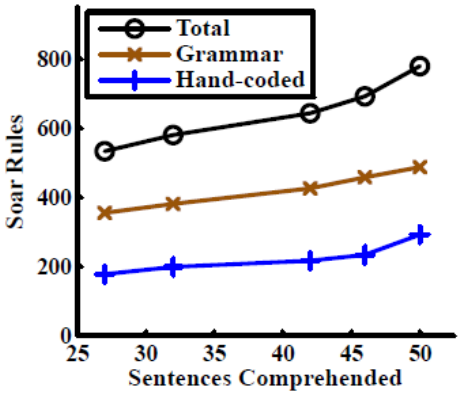


Figure 5: Code growth with knowledge

The number of rules generated from the grammar is much larger than the number of hand-coded ones, and this

proportion grows as the grammar grows. However on the last step, where four questions were added, only 13 ECG items and 29 rules were added to the grammar, while 58 new hand-coded rules were needed. The grammar changes were simple additions, but new ways of attachment, grounding, and formatting were also required. An important issue is whether the number of hand-coded rules plateaus as we extend LUCIA to new constructions.

Experiment 2

To test the generality of the system, we applied LUCIA to a Spanish translation of the same sentences used for Experiment 1, comparing the results to the same gold standard semantic structures used for Experiment 1. The translation was done by the first author, a fluent Spanish speaker, with consultation with a native Spanish speaker. Both have extensive English-Spanish translation experience.

Several linguistic differences needed to be dealt with, in addition to the obvious one of a different vocabulary: adjectives can come either before or after a noun as in *la esfera verde* (*the green sphere*); the morphology of pronouns attached to the end of verbs as in *Levántalo* (*Pick it up*) and *Oriéntate* (*Orient [yourself]*); no equivalent of *then* in *If ... then ...*, although *entonces* could be used with some loss of fluency; word order may be different as in all the example questions; and the meanings of many words, especially prepositions, don't correspond across languages. For example, *on* may be translated as either *en* or *sobre* and *to be* can correspond to either *ser* or *estar*. Spanish also has morphological variation in verb conjugations that English does not have, but that doesn't affect this corpus since everything is in the present tense, all command verbs are in the second person familiar imperative form, and all *to be* verbs are in the third person.

Some new constructions had to be added to handle some of the differences from English. Following these extensions, all 50 sentences were processed correctly. Table 2 shows the relevant code statistics.

Table 2: Experiment 2 statistics

Category	ECG	Rules	Proportion
Common	140	319	36.3%
Spanish-specific	114	263	30.0%
Hand coded	0	296	33.7%
Total	254	878	

Experiment 3

To evaluate the scalability of the system, we took the exact code used for Experiment 1 and ran it on the full original list of 209 sentences. With no additional vocabulary, 110 sentences could not be understood due to 88 unknown words. Of the remaining 99 sentences, the system understood 82. This shows that the system can often process novel sentences that use known words (Item 8).

We then added lexical items for those 88 words, which required adding 113 ECG items that generate 178 Soar rules. With these additions, 92 sentences were understood. This shows that the system can process even more sentences, but also that new constructions must be added to understand many new sentences. It doesn't understand more sentences because the original 209 sentences were chosen to demonstrate a variety of syntactic constructions, which require additional grammatical and semantic knowledge.

Conclusions and Future Work

We set out to evaluate whether LUCIA could provide language comprehension to Rosie in a way that is both useful and cognitively plausible. The above experiments show that it is useful, and that it satisfies, at least partially, the ten cognitive criteria. It does incremental processing that integrates syntax, semantics, and grounding in the perceived world. Its grammar is both hierarchical and compositional. It can eclectically apply all available knowledge at any stage of processing. It has a working repair mechanism and a method for handling lexical ambiguity, although so far these only cover a limited number of cases. Based on Soar assumptions, it comes within a factor of two of real-time processing, and it seems clear how to improve that.

Future work could begin with improving the real-time course of comprehension, adding more robust mechanisms for repair and handling lexical ambiguity, and exploring the correspondence to human limitations that Lewis's (1993) system demonstrates. We can continue on to the much larger challenges of learning grammar and concepts, and using that learning to expand the scope of understandable domains.

Acknowledgments

The work described here was supported by the National Science Foundation under Grant Number 1419590. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the NSF or the U.S. Government.

References

Ball, Jerry, Mary Freiman, Stuart Rodgers and Christopher Myers (2010). Toward a Functional Model of Human Language Processing. Presented as a poster at *32nd Annual Conference of the Cognitive Science Society.* Portland, OR.

Bergen, Benjamin and Nancy Chang (2013). Embodied Construction Grammar. In Thomas Hoffman and Graeme Trousdale, eds, *The Oxford Handbook of Construction Grammar*. Oxford University Press, New York, pp. 168-190.

Bryant, John Edward (2008). *Best-Fit Constructional Analysis*. PhD dissertation in Computer Science, University of California at Berkeley.

Cantrell, Rehj, Matthias Scheutz, Paul Schermerhorn, and Xuan Wu. (2010). Robust spoken instruction understanding for HRI. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 275-282. IEEE.

Coello, Yann and Angela Bartolo, eds. (2013). *Language and Action in Cognitive Neuroscience*. Psychology Press, New York.

Dodge, Ellen Kirsten (2010). *Constructional and Conceptual Composition*. PhD dissertation in Linguistics, University of California at Berkeley.

Feldman, Jerome A. (2006). *From Molecule to Metaphor: A Neural Theory of Language*. MIT Press, Cambridge, MA.

Feldman, Jerome, Ellen Dodge, and John Bryant (2009). Embodied Construction Grammar. In Bernd Heine and Heiko Narrog, eds., *The Oxford Handbook of Linguistic Analysis*. Oxford University Press, New York.

Ferstl, Evelyn C. (1994). The Construction-Integration Model: A Framework for Studying Context Effects in Sentence Processing. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, 289-293.

Fillmore, Charles J. (1976). Frame Semantics and the Nature of Language. In *Annals of the New York Academy of Science, Vol. 280, Origins and Evolution of Language and Speech*, pp. 20-32.

Fillmore, Charles J. (2013). Berkeley Construction Grammar. In Thomas Hoffman and Graeme Trousdale, eds, *The Oxford Handbook of Construction Grammar*. Oxford University Press, New York, pp. 112-132.

Fillmore, Charles J. and Collin Baker (2009). A Frames Approach to Semantic Analysis. In Bernd Heine and Heiko Narrog, eds., *The Oxford Handbook of Linguistic Analysis*, 313-340.

Goldberg, Adele E. (1995). *Constructions: A Construction Grammar Approach to Argument Structure*. The University of Chicago Press.

Goldberg, Adele E. (2006). *Constructions at work: The nature of generalization in language*. Oxford University Press.

Goldberg, Adele E. (2013). Constructionist Approaches. In Thomas Hoffman and Graeme Trousdale, eds, *The Oxford Handbook of Construction Grammar*. Oxford University Press, New York, pp. 15-31.

Hoffman, Thomas and Graeme Trousdale, eds. (2013). *The Oxford Handbook of Construction Grammar*. Oxford University Press, New York.

Kintsch, Walter (1998). *Comprehension: A paradigm for cognition*. Cambridge University Press.

Johnson, Mark (1987). *The Body in the Mind: The Bodily Basis of Meaning, Imagination, and Reason*. The University of Chicago Press, Chicago.

Lakoff, George (1987). *Women, Fire, and Dangerous Things: What Categories Reveal About the Mind*. University of Chicago Press.

Lakoff, George and Mark Johnson (1980). *Metaphors We Live By*. University of Chicago Press.

Lewis, Richard Lawrence (1993). *An Architecturally-based Theory of Human Sentence Comprehension*. PhD dissertation in Computer Science, Carnegie Mellon University.

Lewis, Richard L. and Shavran Vasishth (2005). An Activation-Based Model of Sentence Processing as Skilled Memory Retrieval. *Cognitive Science 29*, 375-419.

Lindes, Peter (2014). *OntoSoar: Using Language to Find Genealogy Facts*. Linguistics master's thesis, Brigham Young University.

Mandler, Jean M. and Cristóbal Pagán Cánovas (2014). On defining image schemas. *Language and Cognition* **0**, 1-23.

Miller, George A. and Philip N. Johnson-Laird (1976). *Language and Perception*. Belknap Press.

Mohan, Shiwali, Aaron H. Mininger, and John E. Laird (2013). Towards an indexical model of situated language comprehension for real-world cognitive agents. *Advances in Cognitive Systems 3, 163-182.*

Newell, Allen (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Schank, Roger C. (1972). Conceptual Dependency: A Theory of Natural Language Understanding. In *Cognitive Psychology 3*, 552-631.

Steels, Luc (2013). Fluid Construction Grammar. In Thomas Hoffman and Graeme Trousdale, eds, *The Oxford Handbook of Construction Grammar*. Oxford University Press, New York, pp. 153-167.

Steels, Luc and Manfred Hild, eds. (2012). *Language Grounding in Robots.* Springer.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Encoding and Accessing Linguistic Representations in a Dynamically Structured Holographic Memory System

Dan Parker (dparker@wm.edu)
Daniel Lantz (dflantz@email.wm.edu)
Computational & Experimental Linguistics Laboratory
College of William & Mary, Williamsburg, VA 23187, USA

Abstract

This paper presents a computational model that integrates a dynamically structured holographic memory system into the ACT-R cognitive architecture to explain how linguistic representations are encoded and accessed in memory. We show that a holographic memory system provides a cognitively plausible and principled explanation for the processing of sentences with negative polarity items (NPIs) like *ever* and *any*. The original ACT-R model fails to capture the full range of human reading times and judgments of grammaticality, whereas the integrated holographic memory model achieves good quantitative fits to human error rates and response latencies. These results provide proof-of-concept for the unification of two independent computational cognitive frameworks.

Keywords: Language processing; Memory; Holographic Reduced Representations; ACT-R

Introduction

A hallmark of human cognition is the ability to encode, access, and process compositional structures (Anderson, 1983; Fodor, 2001; Newell, 1990). A parade case involves language processing. For instance, understanding a sentence in a discourse requires mechanisms for encoding a structured representation of the sentence in memory and for accessing specific pieces of information in that representation later. However, it remains an open question how these mechanisms are neuro-computationally instantiated.

One model that has received much attention is the Lewis and Vasishth (2005) (henceforth LV05) model of sentence processing, realized in the Adaptive Control of Thought—Rational (ACT-R) architecture (Anderson, 1990; Anderson et al., 2004). In the LV05 model, sentence processing is construed as a series of cue-based memory retrievals, subject to similarity-based interference. The model is considered the most precise expression of the working memory retrievals and associated control structures that support language processing, and is commonly used to investigate the timing and accuracy of memory retrieval in sentence comprehension.

An initial success of the LV05 model was that it captured interference effects observed in the processing of linguistic dependencies, such as those involving negative polarity items (NPIs). NPIs are words like *ever* or *any*, which are generally acceptable only in sentences that contain a negative-like word in a syntactically higher position, e.g., ***No bills*** *that the senators supported will* ***ever*** *become law*. Previous work has shown that NPI licensing is highly susceptible to interference in sentences like *The bills that* ***no senators*** *supported will* ***ever*** *become law*, due to the presence of the negative distractor *no* that is in a syntactically irrelevant position (e.g., Drenhaus, Saddy, & Frisch, 2005). Interference manifests as decreased accuracy in judgments of grammaticality and decreased reading time disruptions at the NPI, relative to sentences that lack negation. Vasishth, Brüssow, Lewis, and Drenhaus (2008) argued that such effects are a natural consequence of the error-prone memory retrieval mechanisms embodied in ACT-R. Under this view, encountering an NPI triggers a retrieval for a negative licensor, but the wrong item can be retrieved if it matches some of the retrieval cues.

The LV05 model is able to capture many empirical effects, but there are cases where the model makes the wrong predictions. For instance, Parker and Phillips (2014; submitted) showed that NPI interference effects can be reliably switched on and off, depending on when the memory encoding is probed: interference is observed when the encoding of the licensing context is probed early in the sentence, but the effect disappears when the licensing context is probed from a later point in the sentence (see also Parker, 2014). These findings are unexpected under the ACT-R account, which predicts that interference effects should generalize across contexts, based on the assumption that there is a single set of principles that governs memory access.

Parker and Phillips suggested that the contrasting profiles observed for NPIs reflect untested assumptions about how sentences are encoded in memory. ACT-R assumes that the encoding remains fixed over time. However, the finding that interference can be switched on/off depending on when the encoding is probed suggests that the encoding is not fixed, but rather changes over time, such that the internal items become opaque as candidates for causing interference.

This paper presents a computational model that integrates a holographic memory system (e.g., Plate, 2003) into the ACT-R framework to explain the empirically observed effects that the LV05 model fails to capture. Holographic memory systems assume that the atomic components of a compositional structure are periodically bound together in memory to create a single, unitized encoding for interpretation. A key prediction of our model is that interference effects during linguistic dependency formation should be selective, depending on when the encoding is probed. Modeling results show good quantitative fits to a variety of measures, providing proof-of-concept for the unification of two computational cognitive frameworks.

The research reported in this paper builds on previously published literature on holographic memory models and integrating holographic models with ACT-R. Rutledge-Taylor, Kelly, West, and Pyke (2014) and Kelly, Kwok, and West (2015) have shown that a holographic declarative memory system similar to the one proposed here can be integrated into ACT-R to capture decision-making tasks, the fan effect, and delayed learning. Our model demonstrates that this unified framework can capture more specialized cognitive abilities, such as language processing.

The ACT-R model of sentence processing

ACT-R is a cognitive architecture based on independently motivated principles of memory and cognitive skills, and has been used to study a wide range of cognitive phenomena (Anderson, 1990). The LV05 ACT-R model applies those principles to the specialized task of sentence processing.

In the LV05 ACT-R model, linguistic constituents are encoded as 'chunks' in content-addressable memory, and the syntactic representation of a sentence arises as the consequence of pointers that index the hierarchical relations between chunks. Chunks are encoded as bundles of feature-value pairs. Features include lexical content (e.g., morpho-syntactic and semantic features), syntactic information (e.g., category, case), and local hierarchical relations (e.g., sister, parent). Values for features include symbols (e.g., ±singular, ±animate) or pointers to other chunks (e.g., NP_1, VP_2).

Linguistic dependencies, such as those between an NPI and its licensor, are formed using a general retrieval mechanism that probes all task-relevant chunks in parallel for the left part of the dependency (the target), using a set of retrieval cues. Retrieval cues are derived from the current word, the linguistic context, and grammatical knowledge, and correspond to a subset of the features of the target (Lewis, Vasishth, & Van Dyke). Chunks are differentially activated based on their match to the retrieval cues. The probability of retrieving a chunk is proportional to the chunk's overall activation at the time of retrieval, modulated by decay and interference from other items that match the retrieval cues.

The activation of a chunk i (A_i) is defined as follows.[1]

$$A_i = B_i + \sum_{j=1}^{m} W_j S_{ji} - \sum_{k=1}^{p} PM_{ki} + \epsilon \quad (1)$$

The first term of Equation 1 describes the baseline activation of chunk i, which is calculated according to Equation 2. Equation 2 describes the usage history of chunk i as the summation of all n successful retrievals of i, where t_j is the time since the jth successful retrieval of i to the power of the negated decay parameter d. The output is passed through a logarithmic transformation to approximate the log odds that the chunk will be needed given its usage history. After a chunk has been retrieved, the chunk receives an activation boost, followed by decay.

$$B_i = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right) \quad (2)$$

The second term of Equation 1 reflects the degree of match between chunk i and the retrieval cues. W is the weight associated with each retrieval cue j, which defaults to the total amount of goal activation G available divided by the number of cues (i.e., G/j). Weights are assumed to be equal across all cues. The degree of match between chunk i and the retrieval cues is the sum of the (weighted) associative boost for each retrieval cue S_j that matches a feature value of chunk i. The associative boost that a cue contributes to a chunk that it matches is reduced as a function of the *fan* of that cue, i.e., the number of chunks in memory that match the cue (Anderson, 1974), according to Equation 3.

$$S_{ji} = S - \ln(\text{fan}_j) \quad (3)$$

The third term of Equation 1 reflects the penalty for a partial match between the cues of the retrieval probe and the feature values of chunk i. Partial matching makes it possible to retrieve a chunk that matches only some of the cues, creating the opportunity for retrieval interference (Anderson et al., 2004; Anderson & Matessa, 1997). Partial matching is calculated as the matching summation over the k feature values of the retrieval cues. P is a match scale, and M_{ki} reflects the similarity between the retrieval cue value k and the value of the corresponding feature of chunk i, expressed by maximum similarity and maximum difference.

Lastly, random noise is added to the activation level of chunk i, generated from a logistic distribution with a mean of 0, controlled by the noise parameter s, which is related to the variance of the distribution, according to Equations 4 and 5.

$$\epsilon \sim \text{logistic}(0, \sigma^2) \quad (4)$$

$$\sigma^2 = \frac{\pi^2}{3} s^2 \quad (5)$$

Activation A_i determines the probability of retrieving a chunk, according to Equation 6. The probability of retrieving chunk i is a logistic function of its activation with gain $1/s$ and threshold τ. Chunks with higher activation are more likely to be retrieved.

$$P(\text{recall}) = \frac{1}{1 + e^{(-A_i - \tau)/s}} \quad (6)$$

[1] Readers familiar with ACT-R may notice the non-standard presentation of Equation 1: the sign on the partial match component has been flipped to indicate its penalizing nature.

Activation A_i also determines the retrieval latency T_i of a chunk, according to Equation 7. F is a scaling factor that sets predictions on an appropriate time scale. Chunks with a higher activation value have a faster retrieval latency.

$$T_i = Fe_i^{-A_i} \quad (7)$$

Predictions of the ACT-R model

The LV05 ACT-R model predicts that retrieval for linguistic dependency formation should be subject to interference from non-target or syntactically irrelevant items that match some of the retrieval cues (partial match interference). This prediction is based on the assumptions that retrieval accesses all chunks in parallel and that a partial match between the retrieval cues and a chunk can result in erroneous retrieval of that chunk (see Equation 1). Many studies have shown that this prediction is borne out for a range of dependencies, including subject-verb agreement (Dillon et al., 2013; Wagers et al., 2009; Tanner et al., 2014), anaphora (Parker et al., 2015), case licensing (Sloggett, 2013), and ellipsis (Martin, 2015).

For instance, the LV05 model has been used to explain interference effects observed in the processing of negative polarity items (NPIs). NPIs are words like *ever*, *any*, or *yet*, that can be licensed by a negative-like word in a syntactically higher position. The NPI *ever* in (2a) is licensed because it appears in the scope of the negative phrase *no students*. When negation is absent, (2b), or is in a syntactically irrelevant position, (2c), the NPI is not licensed.

(2) a. *No students* have *ever* passed the test.
b. The students have *ever* passed the test.
c. The students that *no teachers* liked *ever* passed the test.

Previous research has shown that NPI licensing is highly susceptible to interference in sentences like (2c), due to the presence of the negative distractor, e.g., *no teachers*, that is in a syntactically irrelevant position for the purpose of NPI licensing. This effect manifests as decreased accuracy in judgment tasks and decreased reading time disruptions when processing the unlicensed NPI, relative to sentences that lack negation, like (2b).

Vasishth et al. (2008) argued that such effects are a natural consequence of the error-prone retrieval mechanisms embodied in ACT-R. Under this account, NPI licensing is implemented as an item-to-item dependency by retrieving a negative licensor from memory using syntactic and semantic cues, e.g., [+scope], [+negative]. In (2a), retrieval finds an item that matches both cues. In (2b), retrieval fails to find a match to either cue. In (2c), retrieval finds a partially matched item, i.e., a semantically appropriate item in a syntactically irrelevant position. The activation boost from this partial match, combined with stochastic noise, can cause the syntactically irrelevant licensor to be retrieved, spuriously licensing the NPI. Vasishth et al. showed that Equations 1-6 achieve good quantitative fits to both human reading times and judgements of grammaticality.

Challenges for the ACT-R model

The LV05 ACT-R model predicts that interference during NPI licensing should generalize across syntactic environments, since the effect is attributed to error-prone retrieval mechanisms that are engaged whenever an NPI is encountered. However, this prediction is not borne out. Parker and Phillips (2014; submitted) showed that interference effects for NPIs can be reliably switched on/off, depending on when the memory encoding of the licensing context is probed. They manipulated the position of the NPI relative to the potential licensors in sentences like (3), and found contrasting profiles: interference was observed when the NPI appeared early in the sentence, i.e., in the main clause (position 1), replicating previous findings, but not when it appeared later in the sentence, i.e., in the embedded clause (position 2). These effects were shown using both reading time measures and speeded acceptability judgments.

(3) The journalists that no editors recommended ($ever_1$) thought that readers would ($ever_2$) understand physics.

These findings suggest that the interference effects observed for NPIs cannot simply be due to noisy retrieval mechanisms that are engaged whenever an NPI is encountered, as assumed in ACT-R. Furthermore, the effects cannot reflect decay or faulty encoding of the licensing context, since that would predict difficulty in the grammatical conditions, contrary to fact.

Parker and Phillips argued that the contrasting profiles observed for NPIs reflect untested assumptions about how sentence representations are encoded in memory. ACT-R assumes that the encoding of the sentence remains fixed over time. However, the finding that interference effects can be switched on/off depending on when the encoding is probed suggests that the encoding is not fixed, but rather changes over time: at one moment, irrelevant items are transparently accessible via partial matching; but then at a later point in time, those same irrelevant items become opaque as candidates for causing interference.

In the next section, we discuss how such effects are predicted in an alternative, dynamically structured holographic memory system.

Multiple-stage encoding schemes

The LV05 ACT-R model assumes that the encoding of a sentence remains fixed over time. However, this is not a widespread assumption. Many cognitive models, including the entire class of Vector Symbolic Architectures (VSAs), e.g., Tensor Product Models (Smolensky, 1990), Holographic Memory (Plate, 2003), Binary Spatter Codes (Kanerva, 1994), assume that there is a qualitative shift over time in the format of an encoding in memory.

In VSAs, compositional structures are encoded in two stages. When a representation is first encoded, it is equivalent to its subparts, such that the individual features of the representation can be evaluated independently from their position in a structured representation, creating the

opportunity for partial match interference at retrieval. Then, at a later point, those same features may be bound together, creating a single, unitized encoding that is dissimilar to its sub-parts to conserve memory resources. In this state, individual features are no longer independently evaluable, and the representation must exhibit an all-or-none match to the cues of the retrieval probe in order to be recovered, preventing the possibility of partial match interference. This idea of "recoding" is based on Miller's (1956) principle of chunking, which provides a central explanation for how human memory works.

Proposal

An implicit assumption of VSAs is that compositional structures are encoded in multiple stages. VSAs make a distinction between "atomic" representations that are typically randomly generated versus "complex" compositional representations that are constructed from atomic representations. We propose that these two representational stages may be mapped to distinct cognitive processing stages as a principled explanation of the contrasting profiles observed for NPI licensing. Previously, VSA-based cognitive models have not assumed that particular cognitive processing stages are associated with the two representational schemes. However, if the format of the encoding changes over time, as implicitly assumed in VSAs, then we should expect different behaviors at different points in time, depending on when the encoding is probed, as suggested for NPI licensing.

Encoding linguistic structure in multiple stages

In VSAs, the feature-values of a linguistic representation may be encoded as high-dimensional vectors that are recursively bound together by compressing their outer product into a single vector. For instance, in a tensor-product scheme (e.g., Smolensky, 1990), features are bound together in memory by taking the outer product of the vector representations of the features, as shown in (4).

(4) a. Feature vectors
$[+\text{scope}] = [123]; [+\text{negation}] = [\text{abc}]$

b. Tensor-product feature binding
$$\begin{pmatrix}1\\2\\3\end{pmatrix} \otimes \begin{pmatrix}a\\b\\c\end{pmatrix} = \begin{pmatrix}1a & 1b & 1c\\2a & 2b & 2c\\3a & 3b & 3c\end{pmatrix}$$

However, as the structure grows, the size of the code grows exponentially, which is undesirable given the stringent limits on the amount of information that can concurrently occupy working memory (Cowan, 2001). Plate (2003) proposed a solution using Holographic Reduced Representations (HRRs), which rely on circular convolution to bind features together, according to Equation 8.[2] Importantly, the size of the code does not grow as more features are added, since the circular convolution of two n-dimensional vectors using modulo subscripts produces a vector with dimensionality n.

$$t_j = \sum_{k=0}^{n=1} c_k x_{j-k} \quad (8)$$
for $j = 0$ to $n - 1$
(subscripts are modulo-n)

$$\textbf{Binding}_t = [+\textbf{scope}]_c \circledast [+\textbf{negation}]_x$$

$$t_0 = c_0x_0 + c_2x_1 + c_1x_2$$
$$t_1 = c_1x_0 + c_0x_1 + c_2x_2$$
$$t_2 = c_2x_0 + c_1x_1 + c_0x_2$$

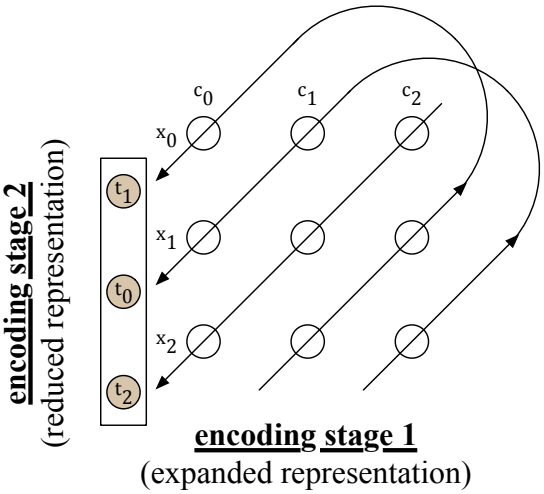


Figure 1. Circular convolution represented as the compressed outer product t of the feature vectors c and x.

Figure 1 shows circular convolution as the ('reduced') outer product t of the feature vectors c and x, corresponding to the linguistic features [+scope] and [+negation] for n=3. Convolution is calculated as the summation of the outer product values along the paths of the lines.

In the uncompressed form (encoding stage 1), individual features c and x are independently evaluable, making the representation susceptible to partial matching. In the 'reduced' form (encoding stage 2), the individual features c and x are no longer independently evaluable, preventing the possibility of partial matching. In this state, the representation must be recovered holistically with an all-or-none match to the cues of the retrieval probe.

Similarity between the retrieval probe p and a memory m measured by their normalized dot product, i.e., cosine similarity, according to Equation 9.

[2] Convolution is the core mathematical operation behind holography, hence the term "holographic".

$$\text{sim}(p,m) = \frac{p \cdot m}{\| p \| \| m \|} = \frac{\sum_{i=0}^{n-1} p_i m_i}{\sqrt{\sum_{i=0}^{n-1} p_i^2}\sqrt{\sum_{i=0}^{n-1} m_i^2}} \quad (9)$$

One concern is that encoding *n*-dimensional bindings using circular convolution can be slow, since convolution calculates the sum of products (convolution with modulo subscripts takes $O(n^2)$ time). Processing can be sped up by performing convolution in the frequency domain with the Fast Fourier Transform, which involves element-wise multiplication, as shown in Equation 10. This process implements circular convolution in $O(n \log n)$ time.

$$[+\text{scope}]_C \circledast [+\text{negation}]_x = f'(f(c) \odot f(x)) \quad (10)$$

The most important property of HRRs, for present purposes, is that the encoding changes such that the internal items become opaque for partial matching with the passage of time. This property could provide a principled explanation for the contrasting profiles observed for NPIs. If the format of the encoding changes over time, as assumed in a holographic memory system, then we should see different behaviors at different points in time, depending on when the encoding is probed.

In the next section, we show how a holographic memory system can be integrated into the LV05 ACT-R model to simulate human reading times and judgments of grammaticality.

Integrating HRRs into ACT-R

A new memory module for the LV05 ACT-R model was developed using HRRs replacing traditional ACT-R chunks with holographic vectors. Holographic vectors retain the same expressive power of the chunks used in the LV05 model, but allow for dynamic changes in the format of the encoding.

To implement HRRs in the ACT-R system, we made the following changes to the original LV05 ACT-R model. First, linguistic feature-value specifications and retrieval cues were encoded as vectors (one dimensional arrays) of *n* numbers, randomly sampled from a normal distribution. For our simulations, n = 10,000. In this format, different feature-value specifications and the corresponding retrieval cues are represented by different patterns in a continuous, high-dimensional space.

In encoding stage 1 (expanded representation), feature-value pairs are superimposed by adding the vectors together to create linguistic chunks (bundles of feature-value pairs, as defined in the original LV05 ACT-R model). Retrieval probe vectors are constructed in the same manner. In this state, the individual features of a chunk are independently evaluable at retrieval and hence susceptible to partial matching, as assumed in the original LV05 model.

In encoding stage 2 (reduced representation), convolution as computed according to Equation 10 is used to bind the vectors representing the feature-value pairs within a chunk. To enable successful retrieval of a chunk, the cues of the retrieval probe must be combined in the same way. In this state, a chunk represents a single, unitized encoding that must exhibit an all-or-none match to the retrieval probe to be recovered, i.e., partial matching is not possible. For present purposes, we assumed that feature binding was triggered upon encountering the main clause verb of a sentence during comprehension. According to Parker and Phillips (submitted), encountering a main clause verb may force the parser to 'wrap-up' and consolidate the encoding of the previous context to conserve memory resources.

Second, we modified the standard ACT-R equation for activation values (Equation 1) to accommodate HRR vectors. Specifically, we substituted cosine similarity, as computed according to Equation 9, for the third term of the standard ACT-R equation for activation value, i.e., the term that computes the penalty for a partial match between the cues of the retrieval probe and the feature values of chunk *i*.

Simulations

We investigated whether the contrasting profiles observed for NPIs would be best captured by the original LV05 ACT-R model or the integrated HRR/ACT-R model. To achieve this, we conducted a side-by-side comparison of the LV05 model with the integrated model, without adjusting key model parameters.

Procedure

Previous implementations of the ACT-R model of sentence processing have included a wide range of modules, including modules for visual information processing, lexical access, memory retrieval, and syntactic parsing (e.g., Lewis & Vasishth, 2005; Vasishth et al., 2008). However, the simulations reported here focus solely on the module for retrieval, and abstract away from the contribution of the peripheral modules by stipulating the chunks in memory and retrievals required to parse a sentence. There are additional processes associated with sentence comprehension that contribute to behavioral measures, but for current purposes, we adopt the standard assumption that the dynamics and output of memory retrieval map monotonically to the behavioral measures of interest (Anderson & Milson, 1989).

We simulated the hypothesized retrievals involved in the key manipulations reported in Parker and Phillips (submitted). Three conditions were simulated, manipulating the presence and location of an NPI licensor (appropriate licensor, irrelevant licensor, no licensor) and the position of the NPI (main clause, embedded clause), based on the sentence structures in (3). For each condition, a schedule of constituent creation times and retrievals was estimated from the reading times reported in Parker and Phillips (submitted). Differences between conditions were modeled only as differences in NPI position and the feature composition of the licensors (±scope, ±negation).

To ensure that the modeling results for the LV05 and integrated HRR/ACT-R model would be directly comparable, all models used the same default parameter settings, following Lewis and Vasishth (2005) and Vasishth

Figure 1. Comparison of predicted and observed interference effects for reading time measures of main clause *ever*$_1$ vs. embedded clause *ever*$_2$.

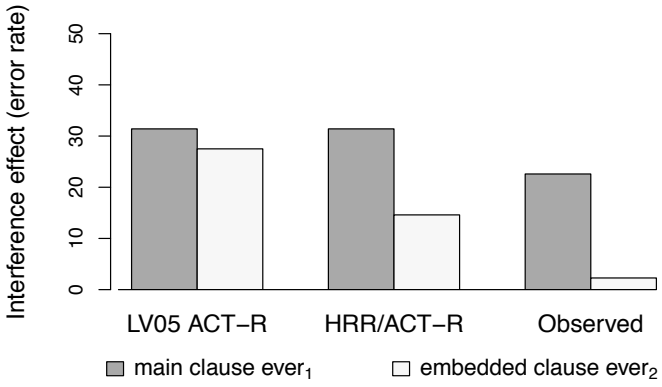


Figure 2. Comparison of predicted and observed interference effects in judgment accuracy for main clause *ever*$_1$ and embedded clause *ever*$_2$.

et al. (2008). The only exception was the scaling parameter F, which was optimized to fit the behavioral time scale (in all models, $F = 0.6$). 5,000 Monte Carlo simulations were run for each condition.

We report two measures of interest: (i) Retrieval error rate reflects the percentage of runs for which the distractor, rather than the target was retrieved. This measure maps monotonically to speeded acceptability judgments, with higher retrieval error rates corresponding to increased rates of judgment errors. (ii) Retrieval latencies reflect the average amount of time it took to retrieve the most probable item, and map monotonically to reading times, with higher latencies corresponding to longer reading times. These measures were used to calculate the predicted interference effect as the difference in predicted error rates and retrieval latencies between the ungrammatical conditions with and without a negative distractor (NPI interference is observed only in ungrammatical conditions). Thus, for predicted error rates, a larger positive value corresponds to a higher rate of interference, reflecting increased rates of acceptance for sentences with a distractor relative to sentences with no distractor. For predicted retrieval latencies, a smaller negative value corresponds to a higher rate of interference, reflecting facilitated processing for sentences with a distractor relative to sentences with no distractor.

We compared the observed interference effects with those predicted by the LV05 model and the HRR/ACT-R model for the reading time measures (Figure 1) and judgment data (Figure 2) reported in Parker and Phillips (2014; submitted).

Simulation results

Across both behavioral measures, the integrated HRR/ACT-R model provided a better fit to the observed data, without adjusting the key model parameters (fit with the HRR/ACT-R model was adjusted $R^2 = 0.79$; fit with the LV05 model was adjusted $R^2 = 0.28$). The LV05 model failed to capture the observed on/off behavior, predicting similar rates of interference across NPI positions. The integrated model, on the other hand, captured the basic contrast between NPI positions, with significantly less interference for embedded clause NPIs (*ever*$_2$).

Although the values predicted by the integrated HRR/ACT-R model did not match the observed data perfectly, the predicted profiles were qualitatively similar to the observed data. We could explore different parameter values to achieve an even better fit with the observed data, but this was not our goal. Rather, our goal was to determine whether the ACT-R model enhanced with a holographic declarative memory system would predict the basic contrasts without adjusting previously fixed parameter values.

The contrasting profiles predicted by the HRR/ACT-R model are consistent with the hypothesis that the contrasting profiles observed for NPIs reflect changes over time in the encoding of compositional representations in memory. After the features of the representation are bound together, the representation must exhibit an all-or-none match to the cues of the retrieval probe, preventing partial match interference.

Conclusion

We presented a computational model that integrates a holographic memory system into the ACT-R model of sentence processing to explain how compositional linguistic structures are encoded and accessed in memory. Modeling results showed that the integrated system is better suited to capture contrasting profiles of interference effects in sentence comprehension, relative to existing models, yeilding a good quantitative fit to data from a variety of behavioral tasks. These results provide proof-of-concept for the unification of two independently developed computational cognitive frameworks, and offer new insights into how humans encode and access compositional representations in memory.

Acknowledgments

The code for the LV05 ACT-R model of sentence processing was generously provided by Rick Lewis. We thank Alan Du for his revisions and additions to this code. We thank Brian Dillon, Dave Kush, Colin Phillips, and Matt Wagers for helpful discussion related to this work.

References

Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology*, *6*, 451-474.

Anderson, J. R. (1983). *The architecture of cognition*. Harvard University Press.

Anderson, J. R., & Milson, R. (1989). Human memory: an adaptive perspective. *Psychological Review*, *96*, 703-719.

Anderson, J. R. (1990). *The Adaptive Character of Though*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*, 1036-60.

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, *24*, 87-185.

Dillon, B., Mishler, A, Sloggett, S., & Phillips, C. (2013). Contrasting intrusion profiles for agreement and anaphora: Experimental and modeling evidence. *Journal of Memory and Language*, *69*, 85-103.

Drenhaus, H., Saddy, D., & Frisch, S. (2005). Processing negative polarity items: When negation comes through the backdoor. In S. Kepser, & M. Reis (Eds.), *Gradience in grammar: Generative perspectives*. NY: Oxford University Press.

Fodor, J. A. (2001). Language, Thought, and Compositionality. *Mind & Language*, *16*, 1-15.

Kanerva, P. (1994). The binary spatter code for encoding concepts at many levels. In M. Marinaro, & P, Morasso (Eds.), *ICANN '94: Proceedings of the International Conference on Artificial Neural Networks*, London, Springer-Verlag.

Kelly, M. A., Kwock, K., & West, R. L. (2015). Holographic Declarative Memory and the Fan Effect: A Test Case for A New Memory Module for ACT-R. In the Proceedings for the 2015 International Conference on Cognitive Modeling (ICCM).

Lewis, R. L., & Vasishth, S. (2005). An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, *29*, 375-419.

Lewis, R. L., Vasishth, S., & Van Dyke, J. A. (2006). Computational principles of working memory in sentence comprehension. *Trends in Cognitive Science*, *10*, 447-454.

Martin, A. E. (2015). Cue-based interference from illicit attractors: ERP evidence from VP ellipsis. Poster at the 2015 Architectures and Mechanisms for Language Processing (AMLAP) Conference.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*, 81-97.

Newell, A. (1990). *Unified theories of cognition: The William James lectures*. Harvard University Press.

Parker, D. (2014). The cognitive basis for encoding and accessing linguistic structure. Unpublished doctoral dissertation.

Parker, D., & Phillips, C. (2014). Time heals semantic illusions, but not syntactic illusions. Talk at the 27th CUNY Conference on Human Sentence Processing.

Parker, D., Lago, M., & Phillips, C. (2015). Interference in the processing of adjunct control. *Frontiers in Psychology*, *6*, 1-13.

Parker, D., & Phillips, C. (submitted). Linguistic illusions and the encoding of compositional representations.

Plate, T. (2003). *Holographic Reduced Representation: Distributed Representation for Cognitive Structures*. CA: CSLI Publications.

Rutledge-Taylor, M. F., Kelly, M. A., West, R. L., & Pyke, A. A. (2014). Dynamically structured holographic memory. *Biologically inspired Cognitive Architectures*, *9*, 9-32.

Sloggett, S. (2013) Case licensing in processing: Evidence from German. Poster at the 26th CUNY Conference on Human Sentence Processing.

Smolensky, P. (1990). Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems. *Artificial Intelligence*, *46*, 159-216.

Tanner, D., Nicol, J., & Brehm, L. (2014). The time-course of feature interference in agreement comprehension: Multiple mechanisms and asymmetrical attraction. *Journal of Memory and Language*, *76*, 195-215.

Vasishth, S., Brüssow, S., Lewis, R. L., Drenhaus, H. (2008). Processing Polarity: How the Ungrammatical Intrudes on the Grammatical. *Cognitive Science*, *32*, 685-712.

Wagers, M., Lau, E. F., & Phillips, C. (2009). Agreement attraction in comprehension: Representations and processes. *Journal of Memory and Language*, *61*, 206-237.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Investigating and Simulating the Effect of Word Fragments as Orthographic Clues in Crossword Solutions

Kejkaew Thanasuan (kejkaew.tha@kmutt.ac.th)
Learning Institute, King Mongkut's University of Technology Thonburi
Bangkok, 10140 Thailand

Shane T. Mueller (shanem@mtu.edu)
Department of Cognitive and Learning Sciences, Michigan Technological University
Houghton, MI 49931 USA

Abstract

A number of models of word structure represent orthography in terms of features indexing individual letters and adjacent letter pairs within the word. This permits word parts to be represented independent of position, but leaves the open question of whether partial letters arranged in multiple clusters (fragments) provide better memory retrieval cues. To answer this, we conducted a study in which expert and novice crossword players completed crossword problems for words of different lengths and with different numbers of letter cues. Although expertise, word length and a number of cues provided strong predictors of accuracy and response times, within each cue/word-length condition, neither the number of word fragments nor the maximum size of word fragment provided a consistent advantage. A computational model using letter pairs as features, but no higher-order representation of orthography, accounted for effects of expertise, word length, and number of cues, and similarly did not produce systematic effects on the number or sizes of fragments. Results suggest that, to a first approximation, letter-pair representations are sufficient to account for the performance in word stem and word fragment completion, crossword, and potentially other word reading/identification tasks.

Keywords: crossword expertise; memory retrieval; orthographic clue; crossword recognitional-based decision-making model

Introduction

Models of reading and word representation in Latin-character languages have often represented orthography in terms of features associated with both individual letters and *letter pairs* (Grainger & Van Heuven, 2003; Thanasuan & Mueller, 2014). For example, the feature-based representation for FORCE would include F, O, R, C, and E, along with *F, FO, OR, RC, CE, and E* (where * indicates a word boundary). This scheme permits representing a sequence without coding the absolute position of a letter within a word, and so it enables similarity-based comparisons to be robust to prefixes and suffixes (ENFORCE, FORCEFIELD) with graded similarity to some grammatical modifications (FORCING). Such models have been successful at accounting for data in a number of reading, memory, and word completion paradigms, but less is known about whether higher-order orthographic representations such as trigrams (or "Wickelphones") are useful, and whether cues involving multiple letter pairs are better than those involving fewer pairs.

Although such representations have often been tested in a general population of readers (who are assumed to have substantial experience with the problems of memory retrieval based on orthographic information), another factor to consider is whether extensive deliberate practice with word-fragment completion changes the level of representation used, or permits better use of multiple word fragments in cueing a correct word. Thus, expert word game players may produce fundamentally different results, showing evidence of different or higher-order representations in word completion tasks in their domains of expertise.

A number of researchers have examined crossword solvers to identify their memory retrieval and problem-solving abilities and strategies. For example, Nickerson (1977, 2011) explored crossword puzzle solving processes relating to lexical memory and categorized daily crossword cues in order to understand information retrieval of the solvers. Moreover, Toma, Halpern, and Berger (2014) compared visuospatial and verbal working memory among college students, crossword and Scrabble experts using a symmetry span task and a reading span task. They found that participants from the two elite groups performed the cognitive tasks better than the novice group, but the results between the two groups were not statistically different.

Mueller and Thanasuan (2013); Thanasuan and Mueller (2014) developed and implemented computational models of crossword solving based on the Recognitional-primed decision making model (RPD; Klein, 1993), the Bayesian Recognitional Decision Model (BRDM; Mueller, 2009), and a computational model of word-stem completion (Mueller & Thanasuan, 2014). The models used data from a database of millions of real crossword clues and answers, and used a letter-pair feature set to represent orthography. Expertise effects were accounted for in terms of speed, strategy, and retrieval fluency, and although the models performed better than novices, they did not achieve as good performance as did experts. This may have arisen in part because of the orthographic representation; a computer performing logical template-based matches of word stems can generally reduce the candidate set substantially more than our representation (see Ginsberg, 2011), which would have improved performance significantly.

Letter Clusters as Orthographic Clues

Typically, a partially-filled answer in a crossword grid is easier to solve than one with no letter cues, in part because it limits the search set in mental lexicon (Nickerson, 2011), and

provides additional cues for retrieval. Thanasuan and Mueller (2014) concluded that although crossword experts used semantic information as a primary constraint, they also relied on orthographic knowledge and visual pattern recognition to complete the puzzles. Supporting this, Mueller and Thanasuan (2013) found that when crossword experts were presented with easy word-stems (i.e. three or fewer missing letters), the accuracy was about 80%, whereas novices were able to complete them correctly only about 40%.

However, those studies did not examine whether clusters of two or more letters were more helpful in memory retrieval than if the same letters are dispersed throughout the clue. Other tasks, such as the cue-facilitated retrieval paradigm, have been used to investigate the role of letter clusters in word completion. For example, Horowitz, White, and Atwood (1968) used this paradigm to determine whether the type of letter cluster (the first, middle or last three-letter clusters of a word) impacted the ability to recall nine-letter words, and found that the first three-letter fragment was the most helpful. Likewise, Dolinsky (1973) compared the cue retrieval process using syllabic and non-syllabic letter clusters as cues. They found that the middle syllabic units were very helpful on word retrieval, but the syllabic clues did not facilitate the recall performance more than the non-syllable fragments. In addition, Goldblum and Frost (1988) studied a cue facilitation effect of letter clusters using a crossword paradigm task. They hypothesized that different structures of sub-lexical units, which include syllable, pronounceable non-syllable, unpronounceable cluster, and nonadjacent letters, might differently influence word retrieval. They found that the small units of syllabic fragments were the best retrieval cue. They also found that the syllables with phonological units such as "- -SEP- - - - -" of a target word "INSEPARABLE" assisted the solvers more than morphemic units (i.e., units linked to meaning) such as "- - - -PAR- - - -" of the same target answer. This suggests that the position of letter clusters within a word and within syllable boundaries may be important.

We suggest that two properties regarding the usefulness of letter clusters are not well understood. The first is whether, for a fixed number of letters in the cue, does their arrangement into clusters impact accuracy of retrieval? For example, if four letters are given for the word "HOUSECAT", they might be "H-U-E-A-", "HO- -EC- -", or "HOUS- - - -". The first has no clusters (sets of adjacent letter pairs), the second has two clusters, and the third has one cluster. It may be that, for either novices or experts, a better or a worse cue is provided when letters are arranged into more clusters. In terms of models of representation, if an advantage exists, this may indicate the use of higher-order representations and require adapting existing models to account for such results. The second property is the maximum length of the cluster—do larger clusters form better cues than smaller clusters? Regarding the previous example, the largest cluster size is one, two and four, respectively. These may differently impact solution probability.

Crossword Solving Model

Previously, we have described a cognitive computational model of crossword solving that accounts for expertise, word length, clue difficulty, and letter-clue effects (Mueller & Thanasuan, 2013; Thanasuan & Mueller, 2014), that we will use in this study. The model simulates crossword answers via two independent routes: semantic and orthographic memory associations (see Figure 1). Mueller and Thanasuan (2013) indicated that the best model representing crossword solving performance was the dual route model with three different conditions for novice, expert and best performance simulations. This model first attempts retrieval via the orthographic route. If it fails, the model searches via semantic associations.

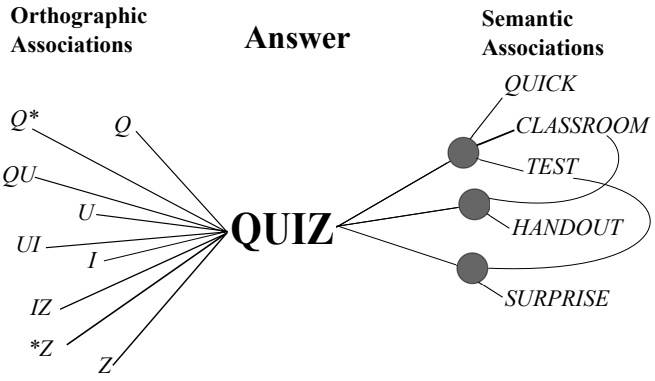


Figure 1: Example of semantic and orthographic routes

The orthographic route model works by representing orthography according to associations between features of a word and crossword answers. Orthographic features that were used include letters, letter pairs, and a distributed length code that helps limit possible answers to words of similar length to the clue. The model was trained on a lexicon of more than 4 million crossword clue answers, and so it has rich associations between these features and existing crossword answers. The representation does not include any higher-order letter clusters (sequences of three or more letters). If there are word fragment effects (depending on size or number of clusters) that cannot be accounted for by the model, this may indicate higher-order representations are needed. The challenge for higher-order representations is that the number of features required scales with the power of the cluster size, making naive representations unmanageable, especially when most of these features never appear in a given language. The alternative would be to begin incorporating syllable representations informed by phonology (see Fudge, 1969; Mueller, Seymour, Kieras, & Meyer, 2003), morphology, or information-theoretic measures, the present study seeks to determine whether this increased complexity is necessary.

To test the model, we investigated the role that word fragments play in crossword puzzle solutions among both experts and novices. A crossword paradigm task was used in which participants were given single clues with partial

letter hints. Although the task requires matching or retrieval from both semantic memory and orthographic memory, we focused on studying the solving mechanisms associated with orthographic clues and word fragments. Furthermore, to assess the role of word fragments on representation, a crossword-solving model based on Mueller and Thanasuan (2013); Thanasuan and Mueller (2014) was adapted to simulate crossword answers and response times. Our approach is to examine whether factors that would be consistent with the use of higher-order representations improve performance, and to demonstrate whether the model shows a similar effect. To do this, we examined whether the size of the largest cluster, and the number of clusters in a clue had an impact in retrieval times or accuracy for both novices and experts.

Experiment

Participants

Eighty-five undergraduate students were recruited from Michigan Technological University (MTU) subject pool as crossword novices. In addition, 113 crossword experts were recruited from online crossword communities. Both groups of participants completed an online crossword study and a demographic survey via web browser. The study protocol was reviewed and approved by MTU Institutional Review Board.

The survey was given to participants at the beginning of the experiment. The novices were 19.94 $\pm$ 1.5 years old in average. Eighty-three percent of them rarely or never solved crossword puzzles, but some of them played other word games such as Scrabble or Words With Friends. The experts were 45.07 $\pm$ 15.99 years old in average. They reported that they have solved crossword regularly for 16.76 $\pm$ 15.35 years and 44 percent of them have participated in crossword tournaments such as American Crossword Puzzle Tournament (ACPT).

Materials and stimuli

Six sets of 15 crossword paradigm task problems were given to participants in this study. The answers were limited to only a set of four, six and eight letter words. In each trial, we gave participants a crossword clue along with some randomly filled letters, as shown in Figure 2. The number of present letters ranged from zero (no letter cue) to only one missing letter (almost a complete answer). They had 15 seconds to solve each trial and only one chance to give an answer.

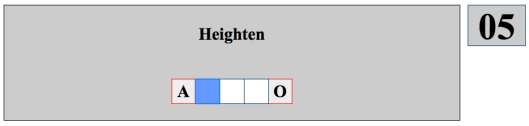


Figure 2: Example of the crossword paradigm task

Experimental Results and Model Simulation

Data from 198 participants were analyzed in this study. Accuracy was assessed by whether the final set of letters were exactly correct after the enter key was pressed to confirm the response, whereas response times were measured from a starting time (when a participant first saw a trial) until the participants hit the enter key. Average accuracy and response time of the crossword experts were 0.76 $\pm$ 0.19 (67 from 90 trials) and 4.82 $\pm$ 2.67 seconds per each trial, respectively. Meanwhile, a mean of success rate of the novices was 0.53 $\pm$ 0.15 (47 from 90 trials) and an average response time was 6.69 $\pm$ 1.88 seconds per each trial. Retrieval times of both experts and novices were estimated from the starting time until the first key was pressed (when the participants typed a first letter to an answer space). An average retrieval time of the novices was 5.73 $\pm$ 1.56 seconds, so the time for the simulations of the Novice model was 0.57 (5.73 divided by a search set size of 10). The time of the experts was 3.13 $\pm$ 1.01 seconds, then the time for the simulation of the Expert model was 0.22 (3.13 divided by a search set size of 25). Average typing speeds of each keystroke of the novices and the experts were 0.35 seconds and 0.22 seconds, respectively.

Figure 3 shows solving performance across the number of letter cues for both success rate and response times. It indicates that the experts performed faster and better than the novices did. Also, both groups of participants performed better when the number of present letters increased. A one-way Analysis of Variance (ANOVA) was used to analyze letter cueing effects, which indicated a significant improvement for both success rates and the response times of both experts and novices as the number of letter cues increased (p-value < 0.05).

Model simulation

To simulate data, we compared two model parameter settings per expertise conditions (Nov=Novice, Exp=Expert), varying the search set size (10, 10, 25, and 50 for models of Nov1, Nov2, Exp1 and Exp2, respectively), recovery (0.5, 3, 25 and 100), and retrieval speed (130 ms, 130 ms, 570 ms, 570 ms). We assigned the same value (10^{-9}) to smoothing orthographic and semantic parameters in order to optimize and balance solving performances of these two routes. These parameters increase chances of getting answers that have been associated to only one item in the memory activation distribution. All other parameters were fixed to the same levels used in previous simulations. The search set size parameter impacts the number of highly-active candidates retrieved during solution; the recovery parameter affects the probability that a response can be generated once a memory trace has been selected, and the retrieval speed affects the time needed to retrieve a candidate and verify whether or not it is correct. Retrieval and typing speeds were determined from the experimental results. Reading speed was taken from Ziefle (1998)'s study regarding an effect of display resolution, which is about 0.33 seconds per word. Then, a solving time of each trial was

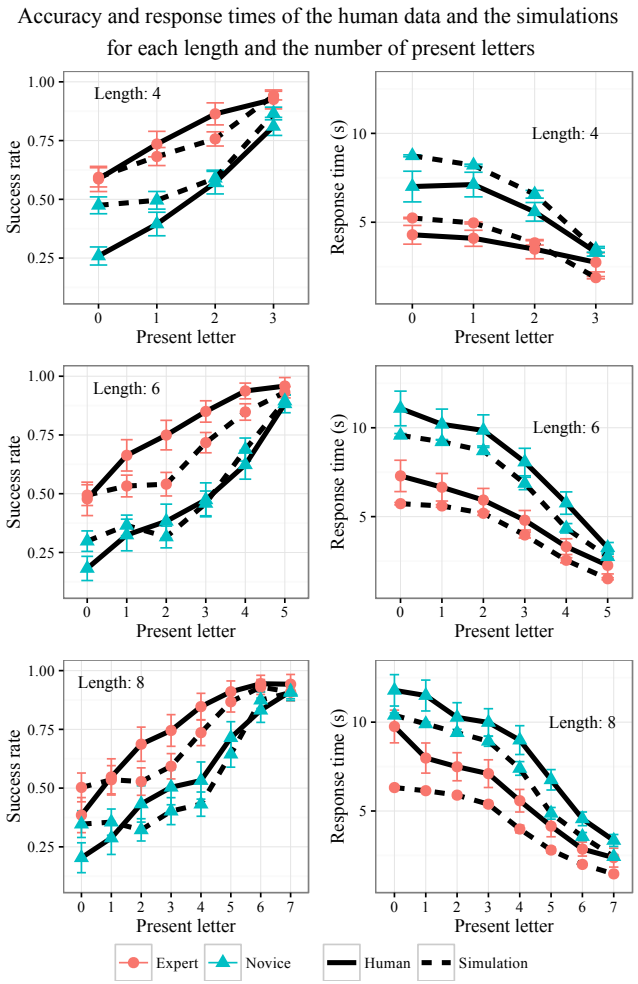


Figure 3: Dots represent means of correct responses and reaction times and error bars indicate 95% confidence intervals

estimated from Equation 1:

$$T_{solving} = cl * t_{reading} + n * t_{retrieval} + wl * t_{typing} \quad (1)$$

where cl is the total number of words in a clue, $t_{reading}$ represents the reading speed, n is the number of candidate answers that the model generates before it gets the first answer that fits the pattern, $t_{retrieval}$ is the retrieval times of the novices and the experts, wl is word length, and t_{typing} is the typing speeds of the novices and the experts, which are 0.35 seconds and 0.22 seconds, respectively. We used two model settings to enable two bracketed models that can account for different levels of expertise.

Table 1 shows simulation results from the crossword play model across the four different settings; two novice models (Nov1 and Nov2) and two expert models (Exp1 and Exp2). The Nov2's success rate and response times were almost the same as the novice data, whereas Exp1 and Exp2 produced success rates and response times closely related to the expert data. Model fits were assessed via Root-Mean-Square Error

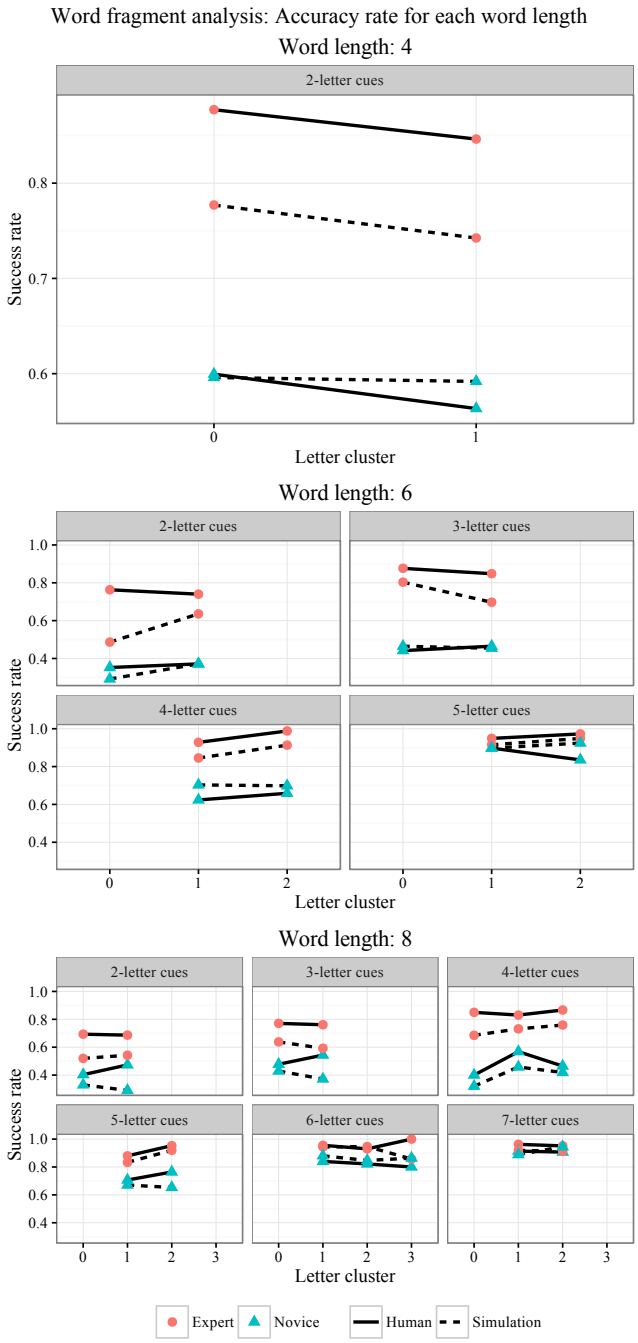


Figure 4: Results for 4, 6, and 8-letter words. Each panel represents a fixed number of letter cues, and the horizontal axis represents the effect as the number of clusters increases.

(RMSE), shown Table 1. The Nov2 model was the best fitting model on both accuracy and response times of the novice data. Meanwhile, Exp2 was the best fitting model in accuracy and Exp1 was the best fitting model in response times of the expert data. However, we chose the Exp1 model to represent the expert data, since an average RMSE of the accuracy and

the response times of Exp1 was less than the other. Moreover, Figure 3 compares the results of Nov2 and Exp1 to the human data.

Table 1: Model results (mean and standard deviation) and Root-Mean-Square Errors (RMSE)

Parameter	Model	Mean (SD)	RMSE	
			Novice	Expert
Acc.	Nov1	0.38 (0.06)	0.17	0.42
	Nov2	0.55 (0.05)	**0.09**	0.26
	Exp1	0.7 (0.04)	0.21	0.1
	Exp2	0.77 (0.04)	0.28	**0.07**
RT (s)	Nov1	4.71(0.46)	1.58	1.9
	Nov2	6.0(0.39)	**1.24**	2.2
	Exp1	3.75 (0.22)	3.83	**1.4**
	Exp2	5.49 (0.37)	2.09	1.83

Note: The bold numbers indicate the smallest value in each performance.

Effects of number of clusters on completion accuracy

The number of letter clusters (groups of two or more adjacent letters) was computed for each orthographic clue. Figure 4 shows means of success rates of each letter cluster for each word length. Each connected line shows how performance changed within each cue number and word length condition as the number of letter clusters increased. Although there were occasional fluctuations, there were no systematic effects of the number of letter clusters on either experts and novices, which was confirmed by a logistic regression and a Chi square goodness-of-fit test (experts: $\chi^2(3) = 6.01, p = .11$ and novices: $\chi^2(3) = 3.22, p = .36$).

The models reasonably replicated human solving abilities (see Figures 3 and 4), although they somewhat underperformed expert performance as a function of number of letters in the cue. On particular word length/number of cue combinations, the model or the humans saw changes with respect to number of clusters and these were sometimes shown in both the model and human data. To the extent that any of these are systematic, they may have stemmed from the way that for any particular word, some combinations of letter clues will do a better job of reducing or eliminating alternative completions, and these combinations may be covered better or worse by clusters of letters. Thus, although the number of letters given improves solution performance significantly, neither the model nor the human data showed systematic effects of this variable on solution accuracy.

Effects of maximum cluster size on completion accuracy

The effects of cluster sizes on accuracy are shown in Figure 5. Similar to the finding with number of clusters, there was little systematic effect of the size of the largest cluster on solution accuracy. Participants occasionally performed better when they were given more adjacent letters, such as for two-letter and four-letter cues of eight-letter words, and in these cases similar effects were seen for both the model and the data. The logistic regression and the Chi-square test were conducted to determine the effects of maximum cluster sizes. The results indicated that the effects were non-significant on accuracy for

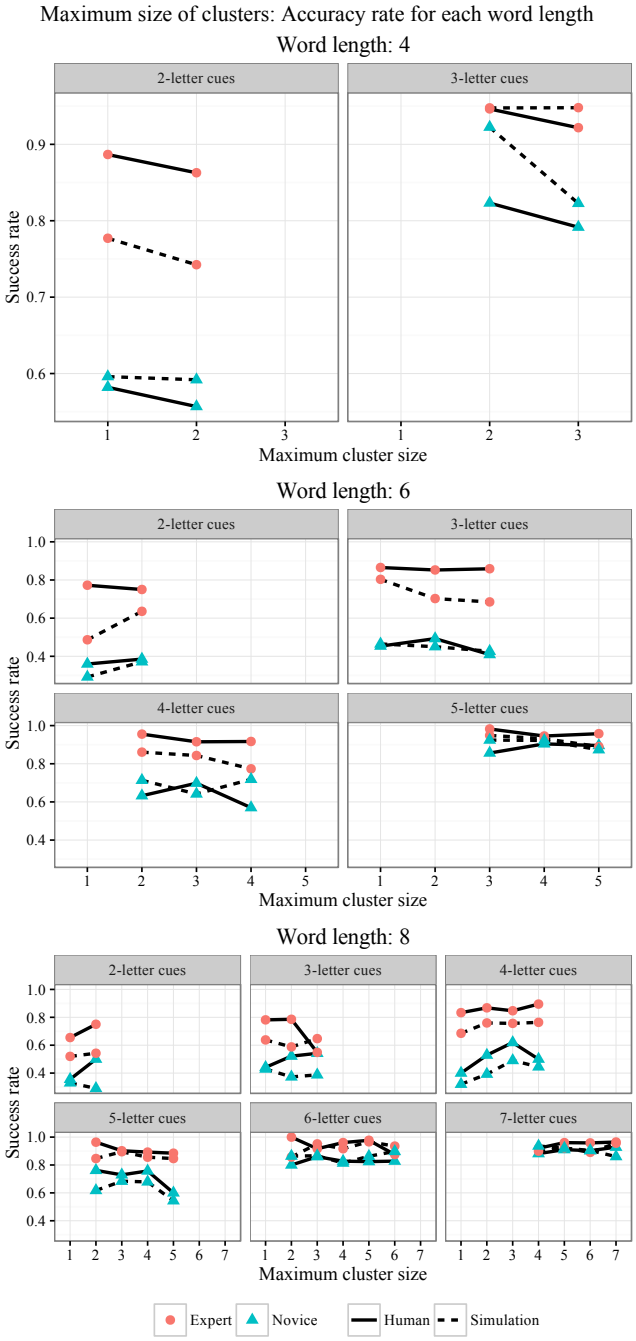


Figure 5: Accuracy by maximum size of cluster. Within each panel (that show a different number of cues) each line shows novice and expert accuracy for different word lengths, as the maximum size of a cluster increases.

novices ($\chi^2(6) = 3.19, p = .78$), and marginally significant for experts ($\chi^2(6) = 12.25, p = .057$). To the extent that the effect exists among experts, it showed that smaller maximum cluster size (which is coupled with greater distribution of isolated letters) tended to lead to better performance, although this was not always the case in each condition.

Discussion

The goals of this study were to investigate the effect of word fragments to determine whether humans enjoyed an advantage of using word fragments that were not predicted by a model using letter-pair representations. We hypothesized that if higher-order orthographic representations were in use, then for a given word length and number of presented letters, when the number or size of word fragments increased, both experts and novices would improve. The findings from the crossword paradigm task suggests that although experts performed crossword solving better than novices did, and the number of letter cues influenced the solving performances on both accuracy and response times, the properties of word fragments we looked at had little impact on performance. Similarly, the model accounted for effects of expertise, word length, and stem size, but showed no systematic effects on these properties of letter clusters. This suggests that, to a first approximation, orthographic models using letter-pairs are still appropriate in representing word retrieval processes.

Nevertheless, we believe that higher-order representations may prove useful in understanding more complex and advanced crossword solving behavior. It may be features associated with morphological, phonological, and syllabically-consistent clusters will both provide substantial advantages for solving, as well as be critical for explaining how long crossword clues are solved. For example, a typical American-style puzzle published in a Saturday New York Times puzzle will have two or more answers that are 15 letters long (ONCEINALIFETIME has been used at least 20 times), and such clues are even more common in cryptic-style crossword puzzles popular outside the United States. More than 5000 such answers have appeared in print[1], and almost all of them are short multi-word phrases. It is likely that the division between composing multiple words, and composing a single word from multiple meaningful lexical units that fit together according to grammatical rules is not as clear as it might seem. Regardless, the representations, processes and mechanisms a model would require to solve multi-word clues (ie., word-level features) would be similar to what would be needed to use syllable or morpheme-level features for single-word clues, and addressing this problem may help understand segmentation in reading, listening, and non-latin languages different rules and practices of segmentation. Consequently, we believe that it may remain useful to consider whether pronounceable clusters, syllables, or morphological units can form features, and to test this in future experiments.

[1] see `http://www.xwordinfo.com/Fifteen`

Acknowledgments

The experiment was conducted while KT was a graduate student at Michigan Technological University.

References

Dolinsky, R. (1973). Word fragments as recall cues: Role of syllables. *Journal of Experimental Psychology*, *97*(2), 272–274.

Fudge, E. C. (1969). Syllables. *Journal of linguistics*, *5*(2), 253–286.

Ginsberg, M. L. (2011). Dr. fill: Crosswords and an implemented solver for singly weighted csps. *Journal of Artificial Intelligence Research*, 851–886.

Goldblum, N., & Frost, R. (1988). The crossword puzzle paradigm: The effectiveness of different word fragments as cues for the retrieval of words. *Memory & Cognition*, *16*(2), 158–166.

Grainger, J., & Van Heuven, W. (2003). Modeling letter position coding in printed word perception. *The mental lexicon*, 1–24.

Horowitz, L. M., White, M. A., & Atwood, D. W. (1968). Word fragments as aids to recall: the organization of a word. *Journal of Exp. Psychology*, *76*(2), 219–226.

Klein, G. A. (1993). A recognition-primed decisions (rpd) model of rapid decision making. In G. A. Klein, J. Orasanu, R. Calderwood, & C. E. Zsambok (Eds.), *Decision making in action* (pp. 138–147).

Mueller, S. T. (2009). A bayesian recognitional decision model. *Journal of Cognitive Engineering and Decision Making*, *3*(2), 111–130. doi: 10.1518/155534309x441871

Mueller, S. T., Seymour, T. L., Kieras, D. E., & Meyer, D. E. (2003). Theoretical implications of articulatory duration, phonological similarity, and phonological complexity in verbal working memory. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, *29*(6), 1353–1380.

Mueller, S. T., & Thanasuan, K. (2013). A model of constrained knowledge access in crossword puzzle players. In R. West & T. Stewart (Eds.), *The 2013 international conference on cognitive modeling (iccm12)* (p. 275).

Mueller, S. T., & Thanasuan, K. (2014). Associations and manipulations in the mental lexicon: A model of word-stem completion. *Journal of Mathematical Psychology*, *59*, 30–40.

Nickerson, R. S. (1977). Crossword puzzles and lexical memory. In S. Dornic (Ed.), *Attention and performance vi* (pp. 699–718). Hillsdale, N.J: Lawrence Erlbaum.

Nickerson, R. S. (2011). Five down, absquatulated: Crossword puzzle clues to how the mind works. *Psychonomic Bulletin & Review*, *18*(2), 217–241.

Thanasuan, K., & Mueller, S. T. (2014). Crossword expertise as recognitional decision making: an artificial intelligence approach. *Frontiers in Psychology*, *5*. doi: 10.3389/fpsyg.2014.01018

Toma, M., Halpern, D. F., & Berger, D. E. (2014). Cognitive abilities of elite nationally ranked scrabble and crossword experts. *Applied Cognitive Psychology*, *28*(5), 727–737.

Ziefle, M. (1998). Effects of display resolution on visual performance. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *40*(4), 554–568.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

ACT-R 3D: A 3D Simulation Environment for Python ACT-R

Sterling Somers (sterling@sterlingsomers.com)
Institute of Cognitive Science, 1125 Colonel By Drive
Ottawa, ON K1S 5B6 Canada

Abstract

In this paper I describe an implementation of a time-synchronous middleware for Python ACT-R and the open-source robotics simulator, MORSE (Echeverria et al., 2012; Echeverria, Lassabe, Degroote, & Lemaignan, 2011), an updated vision system, and an updated motor system, which I collectively call ACT-R 3D. A new vision system and a crude body-model robot was added to the MORSE system to facilitate modelling of affordance-based research on aperture passage (walking through apertures and rotating shoulders as needed). Initial experimental results of shoulder rotation are presented as a proof of concept

Keywords: ACT-R; 3D; affordances; motor control; cognitive modeling;

Introduction

ACT-R is typically used to model psychology experiments in the lab. Relatively little work has been done modelling complex behavior and ACT-R natively has very limited architectural components for motor control. The aim of the project described here was to provide the capacity to model ACT-R models in dynamic, 3D environments. ACT-R 3D amalgamates the robotics simulator, MORSE (Echeverria et al., 2012, 2011), with the Python implementation of ACT-R in a time-synchronous manner. Along with ACT-R 3D is introduced a novel vision system for computer vision in 3D environments, and a motor control system to control a humanoid software robot. As a proof of concept, a model of aperture-passage affordance research is presented. The aperture-passage research has participants walk through apertures (doorways) of various sizes and then measures their degree of shoulder rotation to speculate about cognitive processing while performing these tasks (Higuchi, Seya, & Imanaka, 2012; Stefanucci & Geuss, 2010; Wagman & Malek, 2007; Warren & Whang, 1987).

Background

Although ACT-R has mainly been used for modelling human behavior in psychology experiments, there has been some attempts to model human performance in complex tasks such as driving (Salvucci, Monk, & Trafton, 2009), wayfinding (Trafton & Harrison, 2011), or piloting aircraft (Somers & West, 2013), to name a few. The present work is most similar to ACT-R Embodied (ACT-R/E) which uses ACT-R to control a robot in real-time (Trafton et al., 2012). The aim of the present work was to develop a time-synchronous ACT-R model with a tightly-controlled motor module, sufficient for performing motor control for affordance-based research. In particular, the author used ACT-R 3D to model aperture-passage work by Warren and Whang (Warren & Whang, 1987), where they found that participants rotate relative to the ratio between the width of the aperture and their frontal body width (i.e. shoulder width).

ACT-R/E (Trafton & Harrison, 2011) uses ACT-R as a robot controller and uses the visual system, SECS (Harrison & Schunn, 2003). SECS has three main systems: *visual*, *manipulative*, and *configural.* The visual system uses fiducial and face trackers to provide object identification for video camera images. The manipulative system represents objects as 3D geons (stored in a database) as well as positions and orientation information. The manipulative system also supports spatial transformations, such as rotations in a manner similar to that proposed by Shepard and Metzler (Shepard & Metzler, 1971), supporting motor planning.

ACT-R/E also extends the basic ACT-R motor system. Although somewhat vague in description, Trafton and Harrison (2011) suggests that the motor system maintains real-time limb representations and restricts movements based on muscle groups. It is also suggested by Trafton et al. (Trafton, Harrison, Fransen, & Bugajska, 2009) and by Harrison and Trafton (2010) that motor control is handled external to the ACT-R architecture once ACT-R selects a motor command. For example, given the description by Harrison and Trafton (Harrison & Trafton, 2010), once a representation in the manipulative module is associated with the objects semantic representation, the central production system in ACT-R issues an appropriate grasping command to the robot controller, which carries out the grasp. ACT-R/E has been used in a variety of models. Harrison and Trafton (2010) used ACT-R/E to model response times of grasp actions, Trafton and Harrison (2011) used it to model gaze-following and level-one perspective taking, and SECS (the spatial representation system used in ACT-R/E) was used to model an egocentric navigation task (Harrison & Schunn, 2003).

The present work is a novel implementation comprised of an updated version of Python ACT-R (Python 3), Mobile OpenRobots Simulation Engine (MORSE) (Echeverria et al., 2012, 2011), and a custom middleware responsible for time synchrony and communication between ACT-R and MORSE. The remainder of this paper will describe the system, and a proof-of-concept experimental task (as well as accompanying initial results).

System

The following section describes MORSE and ACT-R 3D.

MORSE

MORSE simulator is a robotics simulator based in Blender[1] (a free, open-source 3D creation studio). MORSE comes with a number of standard sensors and actuators that are pairable to a number of robotic bases. Custom environments, robots, and sensors can be developed in Blender and incorporated into simulations. MORSE is written in Python 3 and the Python library supports full control of robots through Python script. Communication between the robot, sensor, actuators; and the control program is handled through sockets. Models built for ACT-R 3D function as a simulation control script, albeit complex ones. At the time of development, MORSE was at version 1.2. The current version (version 1.4) has a number of advances but should be largely supported by ACT-R 3D.

ACT-R 3D

Python ACT-R Python ACT-R is a re-implementation of ACT-R for the Python programming language (Stewart & West, 2005). Because of the constraints of scripting with MORSE, Python ACT-R was updated to Python 3 for this project. The overall structure and design is largely the same as originally described.

Middleware The middleware between MORSE and ACT-R is designed primarily to support time-synchrony between the ACT-R simulation loop and MORSE. The design of the time-synchrony is inspired by Somers and West (2013), who, as part of a larger project, created a middleware between a popular flight simulator and Python ACT-R. A second major component of the middleware is that it facilitates communication between the ACT-R and MORSE.

In the current implementation, the middleware is designed to run both MORSE and ACT-R at 100 simulated-*Hz,* with no constraints towards real-time simulation. Although a redundant real-time mode is available, in general this project differs from previous work such as ACT-R/E (Trafton & Harrison, 2011) which control robots in real-time. The reasoning behind this is that timing is critical for prediction. Since timing is undoubtably one of the biggest behavioural measures, it is important that timing is as accurate as possible. To avoid any time delays for processing of complex information, that might occur in a computer vision system, or simply even communication time delays, MORSE/ACT-R middleware works in a *tick-tock* fashion. First the ACT-R system *ticks* 10 *ms* of simulation time, sends information get and set requests to the middleware, then, in turn, runs a corresponding 10 *ms tock* in MORSE. At the end of a tick-tock cycle, 10 *ms* of simulation time has elapsed on both simulators. Importantly, any amount of real-time could have elapsed during the tick-tock cycle. The aim is not to have an efficient robot controller but to facilitate prediction of behavioural measures.

Geometric Camera A custom camera class, Geometric Camera, was developed, for MORSE, for the purposes of this project. The intention behind the camera is to provide a single, structured, retinotopic description of the scene from the perspective of the agent.

The camera outputs a dictionary description of the entire scene as visible within its field of view. The dictionary is primarily organized by screen-coordinate *y*-values (where *y* is the vertical plane). Each *y*-value is organized by object label for each object visible to the camera. These labels are not semantically informative; they simply specify different 3D objects in the scene. To each label is associated the object's extension in *x*-values (denoting the object's edges at that *y-value*), and an approximate egocentric distance to the edge. The accuracy of the camera can be passed as parameters, though there can be significant speed/accuracy trade-offs. The overall output is, therefore, a dictionary that describes the egocentric edges of every object visible to the camera. That information is stored privately on the ACT-R side and only accessible through the vision module. Modelers are free to develop their own vision modules to access the visual information to suit their needs.

Vision Module The vision module developed for this project assumes an input from the Geometric Camera (described above). The control of timing, error rates, etc., happen entirely on the ACT-R side.

The agent can only access visual information through requests. However, because the data is largely unstructured, methods for detecting 'features' relevant to the project were developed. A simple *obstacle detector* provides the location of any visible obstacles. Because the task is to walk through a doorway-like aperture an *openings detector* was also developed. The openings detector finds features like doors, holes in walls, or openings between multiple objects. The output of the *openings* detector is not semantically laden. That is, the agent will not be aware of whether the opening is as large as a doorway or as small as a nail hole. The agent model must determine a means of attending to appropriate features. One of the main goals of modelling the aperture-passage task, is that the agent is not semantically informed about its environment. Absolutely no semantic labels are used in the vision system.

The vision module is based on the SOS Vision System (West & Emond, 2002, 2005) in Python ACT-R. Although it is far more complex in the terms of the type of data it deals with, fundamentally the information requests work the same. Requests to the vision system are parameterized in order to filter information. For example, when given a request for an *opening*, chunks that describe the minimum size of the opening are used as parameters for the request. If multiple features match the request (e.g. there are multiple openings), the returned chunk is selected based on a weighted random choice, weighted by a *salience* factor, as described by West and Emond, (2002).

[1] https://www.blender.org/

Motor Module The motor module is one of the features of ACT-R 3D that sets it apart from ACT-R/E (Trafton & Harrison, 2011). While perhaps similar at a functional level (controls limbs, restricts movement), the implementation is novel.

The motor module maintains a hierarchical, symbolic and numerical representation of the body parts (currently only the ones being modelled), that is synchronized with the 3D body-model (robot). For each body part, there are representations of their degrees of freedom. At present the degrees of freedom are represented as max/min values on an axis of rotation. The motor module maintains an internal representation of the maximum and minimum rotations performed by an agent. As the agent moves its body, the maximum and minimum achieved rotations are updated and stored in declarative memory. At a functional level, this memory of body postures can be considered an implementation of body-schema that are theorized to be used in motor planning (Coslett, Buxbaum, & Schwoebel, 2008; J Schwoebel, Coslett, & Buxbaum, 2001; John Schwoebel & Coslett, 2005).

The motor module in ACT-R 3D also includes functionality to provide proprioceptive feedback to estimate 3D body dimensions in a given posture. Evidence for this capability comes from a number of sources (Carello & Turvey, 2004; Stefanucci & Geuss, 2010; Wagman & Taylor, 2005; Warren & Whang, 1987), though anecdotally this ability is intuitively obvious: one does not try to fit their body in a 1 cm^2 hole. We can, at the very least, make crude judgments of body volume, and the above cited suggest the volumetric representations are fairly accurate. How exactly the volumetric representations are achieved are currently beyond the scope of motor module and is implemented, quite simply, with a measure of the agent's bounding box. The bounding box values are associated and stored with the body schema at the time of storage into declarative memory.

Evaluation

ACT-R 3D can be evaluated on its ability to model behavior and make useful predictions of that behavior. This section outlines some preliminary work modelling aperture-passage affordance research.

Aperture-Passage Affordance

The term 'affordance' was first used as a noun by Gibson when he presented an approach to Psychology, Ecological Psychology (1986). An affordance can be thought of as a property (or a set of properties) of the environment[2] that an agent uses in order to determine what actions are available. Action, of course, is a broad category and there is empirical research in support of affordances in a large number of domains including grasping (Tucker & Ellis, 1998), reaching (Carello, Grosofsky, Reichel, Solomon, & Turvey, 1989), stair-climbing (Warren, 1984), a number of sports abilities (see Fajen, Riley, & Turvey, 2009), and relevant here, aperture passage (Warren & Whang, 1987). One of the central concepts in affordance research is Gibson's notion of *direct perception*.

Direct perception is the claim that our actions are not mediated by strong, internal, sensory-based, semantically-laden representations of the environment. Instead, direct perception holds that actions are presented to us in the environment when they are in agreement with our action capabilities. For example, a cup is graspable because its shape and size agrees with the action capabilities of our hands. Realizing an action is a perceptual process and actions are said to be directly perceived.

A common theme in empirical ecological psychology research is to identify a body-scaled unit that is used by the perceptual system to perceive an affordance directly. Researchers identify *pi*-numbers (π) that relate some dimension of the environment (E) with some dimension of the body (A), as a ratio:

$$\pi = E / A.$$

The present work is a first attempt at modelling the classic affordance-passage work by Warren and Whang (1987). Warren and Whang performed a series of experiments aimed to show that aperture passage is directly perceived. In the experiment relevant here, they had participants walk through apertures of different widths, at different speeds (*fast* or *slow*). Participants were grouped according to size: *small* or *large*. Unsurprisingly, smaller agents rotated their shoulders less than large agents when passing through the same aperture. However, when expressed as an aperture-width to shoulder-width ratio (*A/S*), the group differences were eliminated, suggesting that the absolute degree of shoulder rotation is modulated by the *A/S* ratio. Warren and Whang were able to establish a *critical ratio* (1.3) at which, regardless of size, participants would change from maintaining a forward posture, to a posture that included shoulder rotation.

Model

The aperture-passage agent is inspired by work on steering control in a flight-simulator in ACT-R (Somers & West, 2013) in that it uses the SGOMS (West & Nagy, 2007) modelling framework for modelling complex behavior and both bottom-up and top-down visual modules. The simulation environment reflects the experimental setup in Warren and Whang (1987) and is illustrated in Figure 1.

The agent is illustrated in Figure 2. The agent is a low-fidelity humanoid. The robot consists of a rectangular cuboid base mesh with a single armature that, in turn, consists of: a rectangular cuboid pelvic region, a rectangular cuboid torso, a spherical joint between the pelvic region and the torso (not visible) two rectangular cuboid shoulder, two spherical shoulder joints, two cylindrical upper arm segments, a spherical neck joint, a spherical head, and the Geometric Camera. Figure 2 is a polygon reduced version for quicker graphics processing. Each segment of the armature has full degrees of freedom. Although Blender

[2] Their ontological status varies, depending on the author.

does support inverse kinematic solvers for armature control, there are no kinematic solvers associated with any of the armatures in this project. All joint limb control and position representation is controlled by ACT-R 3D, synchronously, as described above. Collision sensors are placed on the shoulder joint, the shoulders, the upper arm segments, and the torso.

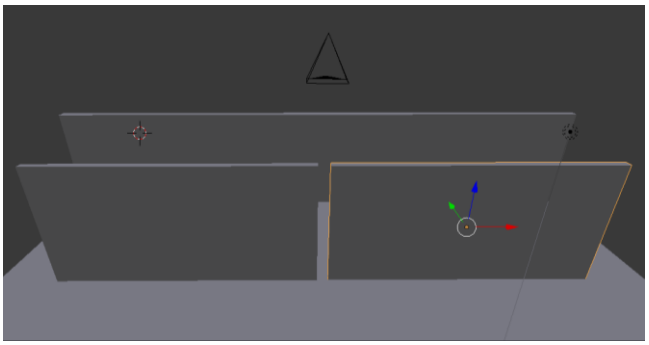


Figure 1: A Snapshot of the modelling environment. Two wall segments create an aperture. The wall segments are moved to create apertures of different sizes.

.

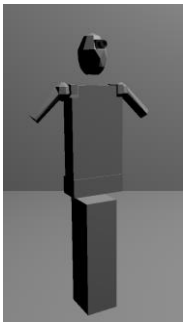


Figure 2: A snapshot of the software robot used in this project. The bottom cuboid represents the agent's legs.

Vision Following the work of Somers and West (2013) a bottom-up vision system was implemented that loops constantly, in this case, looking for obstacles. The bottom-up vision system shares a buffer with the top-down vision system and both also share the main vision module. The buttom-up vision system is implemented using a production system[3] and loops continuously, extracting information from the environment. The bottom-up vision system is constrained by the top-down vision system which is controlled via the central production system. Top-down vision takes priority over the vision module and, as a result, the bottom-up vision system must wait until any top-down requests are complete before it can continue.

[3] Note that Python ACT-R allows for production in modules and this should be thought of as an alternative way of programming modules, not a departure from standard ACT-R.

SGOMS Since an ACT-R implementation of SGOMS has been detailed in (Somers & West, 2013), only a limited description will be provided here. SGOMS --Socio-technical GOMS (Card, Moran, & Newell, 1983)—is a framework for modeling complex tasks and can be implemented in ACT-R. In its ACT-R implementation it uses a hierarchy of buffers that represent the levels of control in a complex task. The hierarchy consists of a planning unit (highest, slowest level of control), the unit task, operators (realized as productions), and methods (compiled productions). The model presented here uses SGOMS to facilitate task interruption.

Design The model can perhaps be best described as going through four phases. The first phase is actually a pre-experiment phase in which the agent stores information about its body size in different postures. Essentially, in this first phase, the agent rotates its shoulders in each direction multiple times, adding the body-schema to declarative memory.

During the second phase, the agent decides whether it can pass through the aperture at all. This is achieved with the vision system. As described above, the *opening* detector is takes dimensions as parameters and filters openings that are too small to accommodate. There are potentially two steps to this phase. The first step involves judging the passability of the opening relative to the agent's present (at the beginning of the simulation) posture. If that fails, a second step is taken. The agent then attempts to recall a body-schema that meets the constraints of the task (e.g. affords walking, involves rotating the shoulders). If a body-schema is recalled (in this case, a posture with complete rotation either to the left or right should be selected), then the dimensions of the agent in that posture is used as a parameter to the visual system's *openings* detector. If there is a match, that means that at maximum rotation, the agent could pass through the aperture. The agent uses the body-schema as a goal for the motor module at time of passage.

During the third phase the agent simply walks towards the aperture. It is assumed that there is very little top-down vision happening. Instead, the bottom-up vision system is cycling for obstacle detection. This phase continues until the agent detects one or both of the walls on either side of the aperture. The obstacle detection causes a task interruption using the SGOMS framework in a similar manner to that described in (Somers & West, 2013). The obstacle alert ultimately results in the fourth phase, rotation control.

Since no data on the kinematic control of rotation is available in the literature, an instantaneous and constant rotation velocity is assumed. Rotation continues until the vision system detects an opening larger than the agent's frontal width (as the agent rotates its shoulders, its frontal width reduces to a point where it is smaller than the width of the opening, resulting in the opening being detected).

Simulations The experimental conditions (including participant sizes) were created as accurately as possible

based of the description by Warren and Whang (1987). There were 5 agents per group condition (*large, small*) and agent sizes were chosen randomly from a normal distribution around the reported (human) means for each group (40.4 *cm, SD* = 2.0 *cm* for small; and 48.4 *cm*, SD = 0.7 *cm* for large). Agents walked at the average speed per group as reported in Warren and Whang: 1.29 *m/s* and 1.61 *m/s* for the normal and fast conditions (respectively) in the small group; and 1.28 *m/s* and 1.77 *m/s* for the normal and fast conditions (respectively) in the large group. Each software participant did 60 trials in each condition, for each aperture.

Because the original data from Warren and Whang's original paper was not available, only a qualitative visual comparison is presented. A visual comparison between the human data in Warren and Whang (1987) is presented in Figures 3 (normal speed) and Figure 4 (fast speed).

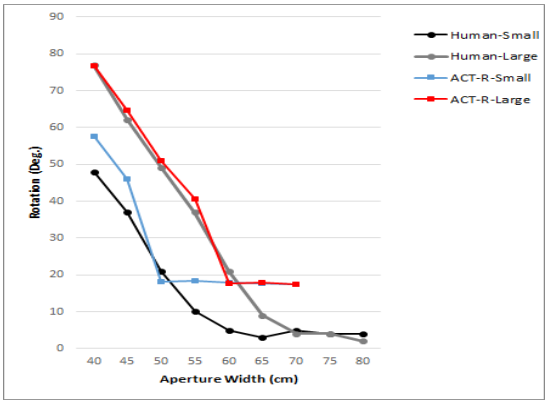


Figure 3: Rotation angle by aperture width for small (blue) and large (red) agents. Gray lines represent human data. Data is from the 'normal' speed condition.

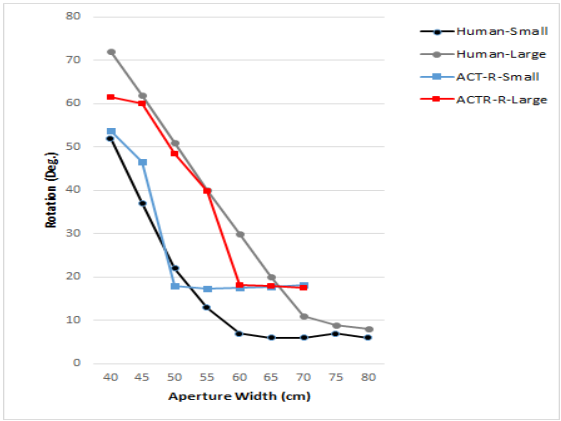


Figure 4: Rotation angle by aperture width for small (blue) and large (red) agents. Gray lines represent human data. Data is from the 'fast' speed condition.

In lieu of comparative statistics, an ANOVA was ran on the model data to see if the main effects described in Warren and Whang's original experiment were replicated. Just as for the human data, participants rotate more for smaller apertures (main effect of aperture), larger participants rotated more than smaller participants (main effect of group) (ps < 0.01). The effect of speed, however, had the opposite effect relative to the findings in Warren and Whang such that the model rotated less when walking at a faster pace (also main effect, $p < 0.01$). However, as illustrated in Figure 5, the group difference in the speed condition is dominated by a single aperture (40 *cm*).

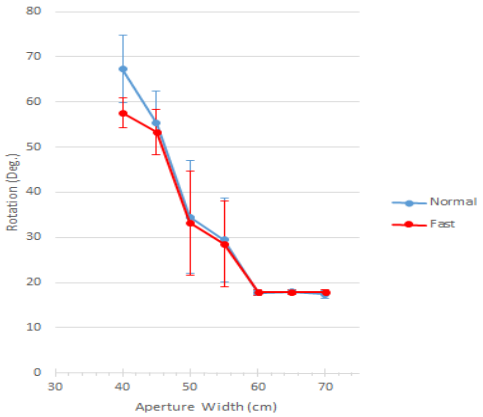


Figure 5: Rotation angle by aperture width for the normal (blue) and fast (red) speed conditions. Error bars represent 95% confidence intervals.

Discussion

ACT-R 3D is designed to support research that involves complex, dynamic environments. Although the model of aperture-passage as presented here has many open questions (some of which have been answered in further analyses, forthcoming), the model, even in its early stages, has offered strong insight to the domain of aperture-passage (though, admittedly it raises more empirical questions than it answers). Forthcoming work will see the model performing similar experiments. Currently it also models aperture passage while carrying an object (Higuchi et al., 2012). Also under development with ACT-R 3D is an embedded driving model, that uses the geometric camera embedded in a simulated car, to drive around a track. Because it uses the Blender simulation engine, a large variety of simulation can potentially be developed in ACT-R 3D for applied cognitive research.

References

Card, S. K., Moran, T. P., & Newell, A. (1983). The keystroke level model for user performance with interactive systems. *Communications of the ACM1*, *23*, 396–410.

Carello, C., Grosofsky, A., Reichel, F. D., Solomon, H. Y., & Turvey, M. T. (1989). Visually Perceiving What is Reachable. *Ecological Psychology*, *1*(1), 27–54. http://doi.org/10.1207/s15326969eco0101_3

Carello, C., & Turvey, M. T. (2004). Physics and Psychology of the Muscle Sense. *Current Directions*

in Psychological Science, *13*(1), 25–28. http://doi.org/10.1111/j.0963-7214.2004.01301007.x

Coslett, H. B., Buxbaum, L. J., & Schwoebel, J. (2008). Accurate reaching after active but not passive movements of the hand: Evidence for forward modeling. *Behavioural Neurology*, *19*(3), 117–125. http://doi.org/10.1155/2008/972542

Echeverria, G., Lassabe, N., Degroote, A., & Lemaignan, S. (2011). Modular open robots simulation engine: MORSE. In *2011 IEEE International Conference on Robotics and Automation* (pp. 46–51). IEEE. http://doi.org/10.1109/ICRA.2011.5980252

Echeverria, G., Lemaignan, S., Lacroix, S., Karg, M., Koch, P., Lesire, C., & Stinckwich, S. (2012). Simulating Complex Robotic Scenarios with MORSE. In *SIMPAR* (pp. 197–208).

Fajen, B. R., Riley, M. A., & Turvey, M. T. (2009). Information, affordances, and the control of action in sport. *International Journal of Sport Psychology*, *40*(1), 79–107.

Gibson, J. J. (1986). *The ecological approach to visual perception*. Hillsdale, NJ: Erlb.

Harrison, A. M., & Schunn, C. (2003). ACT-R/S: Look Ma, no "cognitive map"! In *Proceedings of the Fifth International Conference on Cognitive Modeling* (pp. 129–134). Bamberg, Germany: Universitats-Verlag Bamberg.

Harrison, A. M., & Trafton, J. G. (2010). Cognition for action: an architectural account for "grounded interaction." *Proceedings of the 32nd Annual Conference of the Cognitive Science Society (CogSci 2010)*, 200–205.

Higuchi, T., Seya, Y., & Imanaka, K. (2012). Rule for Scaling Shoulder Rotation Angles while Walking through Apertures. *PLoS ONE*, *7*(10), 1–8. http://doi.org/10.1371/journal.pone.0048123

Salvucci, D. D., Monk, C. A., & Trafton, J. G. (2009). A Process-Model Account of Task Interruption and Resumption: When Does Encoding of the Problem State Occur? In *Proceedings of the Human Factors and Ergonomics Society 53rd Annual Meeting* (pp. 799–803). Santa Monica, CA: Human Factors and Ergonomics Society. Retrieved from http://pro.sagepub.com/content/53/12/799.short

Schwoebel, J., & Coslett, H. B. (2005). Evidence for Multiple, Distinct Representations of the Human Body. *Cognitive Neuroscience*, *17*(4), 543–553.

Schwoebel, J., Coslett, H. B., & Buxbaum, L. J. (2001). Compensatory coding of body part location in autotopagnosia: Evidence for extrinsic egocentric coding. *Cognitive Neuropsychology*, *18*(4), 363–381.

Shepard, R. N., & Metzler, J. (1971). Mental Rotation of Three-Dimensional Objects. *Science*, *171*(3972), 701–703. http://doi.org/10.1126/science.171.3972.701

Somers, S., & West, R. (2013). Steering Control in a Flight Simulator Using ACT-R. In R. L. West & T. C. Stewart (Eds.), *Proceedings of the 12th International Conference on Cognitive MOdelling*. Ottawa: Carleton University.

Stefanucci, J. K., & Geuss, M. N. (2010). Duck! Scaling the height of a horizontal barrier to body height. *Attention, Perception & Psychophysics*, *72*(5), 1338–49. http://doi.org/10.3758/APP.72.5.1338

Stewart, T. C., & West, R. L. (2005). Python ACT-R: A New Implementation and a New Syntax. In *12th Annual ACT-R Workshop*.

Trafton, J. G., & Harrison, A. M. (2011). Embodied Spatial Cognition. *Topics in Cognitive Science*, *3*(4), 686–706. http://doi.org/10.1111/j.1756-8765.2011.01158.x

Trafton, J. G., Harrison, A. M., Fransen, B. R., & Bugajska, M. D. (2009). An embodied model of infant gaze-following. *International Conference of Cognitive Modeling*, (2003).

Trafton, J. G., Hiatt, L. M., Harrison, A. M., Tamborello II, F. P., Khemlani, S. S., & Schultz, A. C. (2012). ACT-R/E: An embodied cognitive architecture for Human-Robot Interaction. *Journal of Human-Robot Interaction*, *1*(1), 78–95.

Tucker, M., & Ellis, R. (1998). On the relations between seen objects and components of potential actions. *Journal of Experimental Psychology. Human Perception and Performance*, *24*(3), 830–46.

Wagman, J. B., & Malek, E. A. (2007). Perception of Whether an Object Can Be Carried Through an Aperture Depends on Anticipated Speed. *Experimental Psychology (formerly "Zeitschrift Für Experimentelle Psychologie")*, *54*(1), 54–61. http://doi.org/10.1027/1618-3169.54.1.54

Wagman, J. B., & Taylor, K. R. (2005). Perceiving Affordances for Aperture Crossing for the Person-Plus-Object System. *Ecological Psychology*, *17*(2), 105–130. http://doi.org/10.1207/s15326969eco1702_3

Warren, W. H. (1984). Perceiving affordances: visual guidance of stair climbing. *Journal of Experimental Psychology. Human Perception and Performance*, *10*(5), 683–703.

Warren, W. H., & Whang, S. (1987). Visual guidance of walking through apertures: body-scaled information for affordances. *Journal of Experimental Psychology. Human Perception and Performance*, *13*(3), 371–83.

West, R. L., & Emond, B. (2002). SOS: A Simple Operating System for modeling HCI with ACT-R. In *Seventh Annual ACT-R Workship*. Pittsburg, PA.

West, R. L., & Emond, B. (2005). The Environment as Theory : An Example Using the ACT-R / SOS Environment. *Proceedings of the Sixth International Conference on Cognitive Modeling*, 398–399.

West, R. L., & Nagy, G. (2007). Using GOMS for Modeling Routine Tasks Within Complex Sociotechnical Systems : Connecting Macrocognitive Models to Microcognition. *Joural of Cognitive Engineering and Decision Making*, *1*(2), 186–211. http://doi.org/10.1518/155534307X232848.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Efficient Parameter Estimation of Cognitive Models for Real-Time Performance Monitoring and Adaptive Interfaces

Christopher R. Fisher (christopher.fisher.27.ctr@us.af.mil)
711^{th} Human Performance Wing, Air Force Research Laboratory, Wright-Patterson Air Force Base, OH, USA

Matthew M. Walsh (mmw188@gmail.com)
Tier1 Performance Solutions, Covington, KY, USA

Leslie M. Blaha (leslie.blaha@pnnl.gov)
Visual Analytics, Pacific Northwest National Laboratory, Richland, WA, USA

Glenn Gunzelmann (glenn.gunzelmann@us.af.mil) & Bella Veksler (bellav717@gmail.com)
711^{th} Human Performance Wing, Air Force Research Laboratory, Wright-Patterson Air Force Base, OH, USA

Abstract

Real-time monitoring provides an opportunity to examine the temporal dynamics of cognition, predict future behavior, and implement adaptive interfaces designed to mitigate declining performance. However, real-time monitoring poses a practical challenge because current parameter estimation methods are prohibitively slow, and real-time monitoring requires parameters to be estimated repeatedly as new data arrive. We developed a real-time parameter estimation method that involves storing pre-computed predictions in a distributed array and using it as a large look-up table. We term this method the Pre-computed Distributed Look-up Table (PDLT). We applied the PDLT to an ACT-R model of the psychomotor vigilance test. PDLT estimates model parameters in just over 1 second with accuracy comparable to that of a much slower simplex method. We discuss methods for reducing the volatility of parameter estimates and the potential to scale up the PDLT method to more complex models and tasks.

Keywords: ACT-R; Psychomotor Vigilance Test; Fatigue; Parameter Estimation; Real-Time Monitoring

Introduction

Performance often declines as a result of various cognitive modifiers or stressors (fatigue, load, etc.; Gluck & Gunzelmann, 2013). In contrast to raw performance metrics, such as mean reaction time, cognitive models provide a principled method for understanding and predicting performance decrements because they formalize the cognitive processes that underlie performance. Moreover, because cognitive models specify the mechanisms underlying performance, predictions from cognitive models are more likely to show greater generalization across tasks compared to performance metrics. For these reasons, cognitive models have the potential to play a prescriptive role in the development of adaptive interfaces designed to mitigate declining performance. When a performance decrement is predicted, an adaptive interface may compensate, for example, by increasing the salience of cues, recommending a multitasking strategy, or prescribing a period of rest (e.g., Rouse, 1988).

Standard approaches to model fitting are ill-suited for implementing cognitive model-based adaptive interfaces, as they often involve pooling data across subjects and are conducted post-hoc. The practice of pooling data neglects multiple sources of variation in performance, resulting in poor individual user prediction and decreased efficacy of adaptive interfaces. For example, performance varies from individual to individual, as well as during a task performed by the same individual. Moreover, performance decrements might be attributable to several causes rather than a single cause, leading to additional variation across individuals and situations. Post-hoc model fitting is problematic because it precludes the ability to predict and mitigate cognitive modifiers.

In contrast, real-time monitoring provides the opportunity to examine variation in cognitive activity attributable to individual differences and changes caused by cognitive modifiers as they unfold (Wilson & Russell, 2003). Once parameterized, a cognitive model can make individualized performance predictions, which in principal could be used to anticipate future performance breakdowns. Individualized interventions could be administered to effectively mitigate performance decrements. With the exception of very simple models in intelligent tutoring systems (Corbett & Anderson, 1994), this application of cognitive models has not been realized because of computational limitations in model fitting.

An important challenge to overcome in using real-time monitoring is increasing the speed with which model parameters can be updated with incoming data. The primary reason for the reliance on post-hoc model fitting is practical: many cognitive models require computationally intensive simulations to generate predictions from a given set of parameters, and the best fitting parameters are often not known in advance. Thus, the parameters must be calibrated to the data using an exhaustive grid search or a search algorithm. In both cases, the process of calibrating the model to data can require hundreds or even thousands of processor hours on High Performance Computing resources (Harris, 2008), thereby rendering real-time monitoring impractical.

As a first step toward using real-time monitoring and adaptive interfaces, we developed a real-time parameter estimation method for quickly and accurately obtaining maximum likelihood estimates for simulation-based cognitive models. We termed this method Pre-computed Distributed Look-up Table (PDLT). As its name implies, this method functions much like a large look-up table in which predictions are pre-computed

and stored for later use, so they can be evaluated in parallel during a real-time monitoring task. As such, nearly all of the computational burden is offloaded prior to the experiment, allowing for the possibility of real-time monitoring.

In the following section, we provide a detailed description of the PDLT method. Next, we describe two simulations using an ACT-R model of the psychomotor vigilance test. The first simulation assessed the speed of the PDLT with different sampling resolutions and amount of data. The second simulation was designed to compare the speed and accuracy of the PDLT to the simplex method.

Pre-Computed Distributed Look-up Table

A wide variety of methods exist for parameter estimation, including grid search, the simplex algorithm (Nelder & Mead, 1965), and a large class of algorithms based on principles of biological evolution (Gen & Cheng, 2000). Each method can be defined as a point in trade-off space in which speed is sacrificed for accuracy. However, even with the benefit of distributed computing, these methods are prohibitively slow for real-time monitoring. The computational bottleneck can be attributed to (1) the reliance on computationally intensive simulation, and (2) the repeated evaluation of similar portions of the parameter space. Together, these factors greatly limit the speed of parameter estimation.

Our goal in developing the PDLT method was to minimize the speed-accuracy trade-off inherent in the aforementioned methods. Our solution was to pre-compute the predictions associated with plausible parameter combinations and store these predictions in a distributed look-up table for later use. A clear advantage of this approach is that it drastically reduces the computational burden during a real-time monitoring task. A related advantage is that pre-computation allows the PDLT to easily scale up to more complex models.

The PDLT method, displayed schematically in Figure 1, entails the following three steps:

Step 1. As represented by the top distribution, the first step is to define a distribution over the allowable parameter space Θ, from which parameter combinations θ are sampled. For illustrative simplicity, we display a unidimensional distribution over the parameter space. However, a distribution of any dimensionality can be used. The purpose of the parameter distribution is to sample plausible values of θ with a high probability. The distribution of parameters should be informed by a combination of theory and parameters obtained from previous studies. The choice of sampling resolution will depend on the the desired speed and accuracy as well as the goals of the application. Accuracy depends on the selected sampling resolution relative to the spread of the parameter distribution. A more granular sampling resolution will suffice if the goal is to identify qualitative patterns of behavior or cognition. However, a higher sampling resolution might be required if the goal is to generate predictions because small quantitative errors may become larger and larger the further out the predictions are extended.

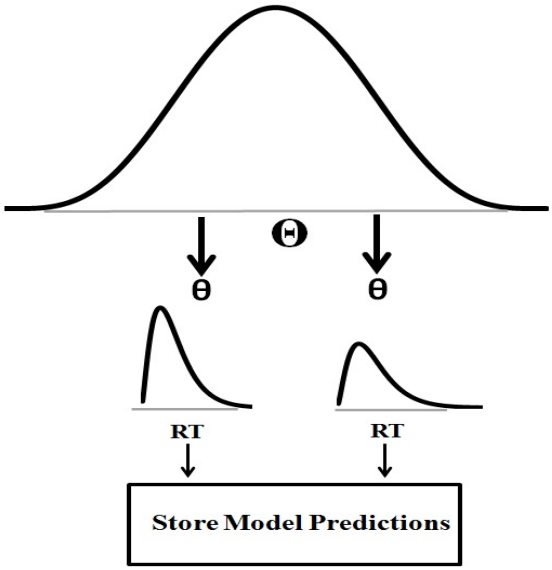


Figure 1: A schematic of the PDLT method. At the top, a distribution is defined over the parameter space. In the middle, predictions are simulated from the model using specific parameters. At the bottom, the predictions are summarized with a statistic and stored in a distributed look-up table.

Step 2. Simulated data are generated from the model for each θ, and predictions are summarized with a statistic. The middle level of Figure 1 indicates that the behavioral output of our model is a reaction time (RT) distribution. In general, a variety of statistics can be used, including means, proportions, quantiles, and kernel density functions.

Step 3. The statistics are stored for later use in any data structure necessary for a given application. Once the distributed arrays are stored, steps 1 and 2 can be omitted in future applications unless the parameter distribution requires modification due to poor model fit.

Evaluation

We performed two simulations using an ACT-R model of the psychomotor vigilance test (PVT). In the first simulation, we examined the speed of the PDLT method as a function of sampling resolution and the number of trials to which the model was fit. In the second simulation, we validated the PDLT method with a parameter recovery study comparing its accuracy to that of the simplex method. Note that although we demonstrate the PDLT on the PVT, the method will extend to most simulation-based cognitive models and tasks.

Common Methods

Hardware

Simulations were performed on a cluster of four Mac Pro computers, each with 16 3.0 GHz cores and 32 GB RAM.

Software

Parameter recovery simulations were programmed in Julia 0.4, a fast, high-level, scientific computing language available via an open source MIT license (Bezanson et al., 2014).

Task

We used the PVT (Dinges & Powell, 1985) to assess the adequacy of PDLT for real-time parameter estimation. The PVT is an ideal task for testing real-time parameter estimation as a proof of concept because of its relative simplicity and its sensitivity to changes in fatigue occurring within 2 to 5 minutes (Loh et al., 2004). The PVT is a simple 10-minute detection task that requires subjects to respond as quickly as possible once the stimulus appears on a test device. The stimulus appears after a random 2-10 sec inter-stimulus interval (ISI). RT distributions from the PVT are empirically rich and are often used to examine the effects of sleep deprivation and fatigue on sustained attention (Lim & Dinges, 2008). A hallmark of fatigue is an increase in the mean, variance, and skew of the RT distribution. Furthermore, an increase in both false starts (RTs before or within 150 ms of stimulus onset) and lapses (RTs $>$ 500 ms) is typically observed as fatigue increases.

Model

We used an ACT-R model of the PVT (Gunzelmann et al., 2009) to demonstrate the use of the PDLT method for two reasons. First, the PVT is of practical relevance for real-time monitoring as it is sensitive to fatigue due to sleep deprivation (Walsh et al., 2014) and time on task (Veksler & Gunzelmann, Under Review). Second, the model has been validated as a plausible account of fatigue decrements in the PVT (Gunzelmann et al., 2015). The model posits that the PVT performance can be characterized with three productions: (1) wait for the stimulus to appear; (2) attend to the stimulus; and (3) respond to the stimulus. During each production cycle, a production is selected stochastically according to partial matching between production rules and the internal and external conditions represented by the model. This partial matching between productions and conditions allows false starts to occur with some low probability. Two parameters are principally responsible for the probability of production selection–a utility scalar (U_S) and a utility threshold (U_T). Eq. 1 provides a formal representation of the production utility:

$$U_{ij} = U_S(U_i - MMP_{ij}) + \varepsilon \tag{1}$$

U_{ij} is the utility of production i in state j, U_S is the utility scalar, U_i is the stored utility for production i, MMP_{ij} is the mismatch penalty for production i in state j, and ε is logistically distributed noise. A value of U_{ij} is assigned to matches, and a value of 0 is assigned to mismatches, yielding a symmetrical payoff matrix. An important feature of the payoff matrix is that the mismatch penalty ensures that the incorrect production is enacted with low probability. The production with highest utility is selected and enacted if its utility exceeds the utility threshold, U_T:

$$\text{Production} = max(U_{ij}) \text{ if } max(U_{ij}) > U_T \tag{2}$$

In the event that no production utilities exceed the utility threshold, no production is enacted, resulting in a microlapse. When a microlapse occurs, utility in Eq. 1 is decremented by a scalar, FP_{dec}: $U_S = U_S \cdot FP_{dec}$. The likelihood of microlapses increases in subsequent production cycles. This leads to behavioral lapses (RTs $>$ 500 ms), and generally lengthens the right tail of the RT distribution. The difference between U_S and U_T, denoted as *Diff*, is an important indicator of fatigue. Relatively low values of *Diff* are associated with greater fatigue. The parameter *cycle time* controls the duration of conflict resolution at the start of each production cycle. The summed duration of each of these processes constitutes the observed RT. Importantly, the duration of each process is stochastic, giving rise to the characteristically right-skewed RT distribution. In summary, the ACT-R model uses four free parameters: U_S, U_T, FP_{dec}, and *cycle time*.

PDLT Specification

The first step in implementing the PDLT method is to define a distribution over the plausible parameter space. We opted for multivariate Gaussian distributions because they account for the central tendency and covariance structure in the empirical parameter estimates, thereby ensuring that unlikely parameter combinations are not needlessly evaluated (see Table 1). The model parameters were derived from reported parameters of related models (Walsh et al., 2014; Veksler & Gunzelmann, Under Review). The parameters were based on well-rested and fatigued subjects to capture a wide range of behavior. One multivariate Gaussian distribution was based on 33 well-rested subjects (Walsh et al., 2014; Veksler & Gunzelmann, Under Review) and the other was based on 13 subjects who underwent 72 hours of sleep-deprivation (Walsh et al., 2014).

We used a high sampling resolution of 150,000 parameter combinations to achieve a relatively high degree of accuracy. During parameter estimation, this required about 2.90 GB of RAM on the primary core and about 325 MB for the remaining cores. The parameter combinations were evenly sampled from multivariate Gaussian distributions associated with well-rested subjects and sleep-deprived subjects.

We used kernel density functions as our fit statistic because they can be used to find maximum likelihood estimates, which have desirable statistical properties, such as consistency and efficiency (Van den Bos, 2007). A kernel density function uses empirical or simulated data to approximate a continuous probability density function. The estimation process involves weighting existing data points as a decreasing function of distance from the target point.

For each parameter combination, a kernel density function was estimated from 64,000 simulated trials. A large number of simulations were used with a small bandwidth (.008) to prevent distortion of the distribution where the false starts end and alert responses begin. Distortion can otherwise occur because the kernel density estimator will be weighted heavily

Table 1: Mean and standard deviations (SD) of parameters used in the PDLT method.

	Utility	Threshold	FP_{dec}	Cycle Time	Diff
Well-Rested	5.86 (1.27)	5.04 (1.00)	.99 (.01)	.05 (.01)	.83 (.63)
Fatigued	3.50 (.51)	3.73 (.50)	.98 (.01)	.04 (.01)	-.23 (.16)

toward the more frequent alert RTs. An object was stored in a distributed look-up table, so the kernel density could be efficiently reconstructed in real time. We used the DistributedArrays package in Julia to spread the computational burden over a cluster of four desktop computers. The look-up table required approximately 35 hours to generate.

Simulation 1

Simulation 1 examined the speed of the PDLT method with different sampling resolutions and number of RTs. The size of the look-up table was varied from 1 to 150,000 in 10 equally spaced increments. The number of trials was either 100 or 1,000. The results were averaged over 100 repetitions for each combination of sampling resolution and number of RTs to produce a stable estimate.

Results

Figure 2 shows the mean completion time for the PDLT method as a function of sampling resolution and number of RTs. Across all combinations, the mean completion time was well below the minimum ISI of 2 seconds in the PVT, indicating that the PDLT method is suitable for trial-by-trial monitoring of task performance on the PVT.

Results indicate that increasing the sampling resolution produces a modest linear increase in completion time. Additionally, increasing the number of trials results in a small overall increase in completion time. By comparison, the simplex method requires about 1-4 minutes to fit the model using the same hardware and software, depending on the number of iterations, number of starting points, and number of simulations per evaluation.

Simulation 2

Simulation 2 was a parameter recovery study designed to compare the PDLT to the simplex method. Parameter recovery involves generating simulated data from a model with known parameters and fitting the model to the simulated data to assess the accuracy of the parameter estimates.

A total of 100 parameters were selected as the ground truth. Half were from the multivariate Gaussian distribution for well-rested subjects, and half were from the sleep-deprived subjects' multivariate Gaussian distribution (Table 1). For each parameter combination, 50 trials were simulated with the ACT-R model. The ACT-R model was fit to these simulated data sets using the PDLT and simplex methods.

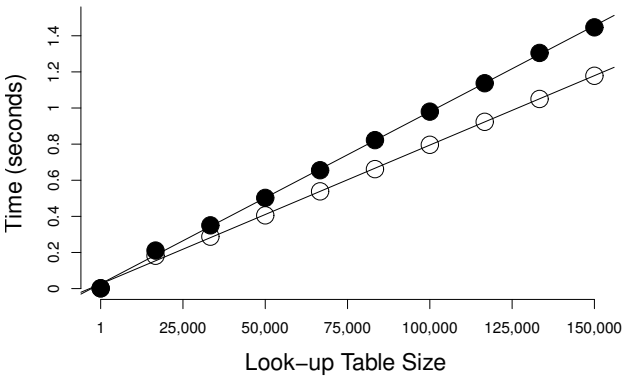


Figure 2: Mean completion time as function of look-up table size and number of trials. Filled circles denote 1,000 trials and unfilled circles denote 100 trials.

Simplex Method

We compared the PDLT to the simplex (Nelder & Mead, 1965) because it is commonly used, widely available, and applies to nonlinear models without tractable derivatives, such as ACT-R. Other algorithms will produce similarly long completion times due to their reliance on simulation. To ensure robustness to local maxima, we computed the likelihood of 50 candidate starting points and initialized the simplex with the three with the highest likelihood. Forty-nine candidate starting points were sampled from the multivariate Gaussian distributions, and the remaining candidate starting point was the best-fitting parameters from the PDLT for each corresponding data set. Initializing the simplex algorithm in this manner provides a rigorous test of the PDLT method because the simplex method has the benefit of fine-tuning the PDLT estimates in addition to using multiple starting points.

Next, we ran the simplex on three starting points with the highest likelihood. For each starting point the algorithm performed 100 iterations before recording the best-fitting parameters for that particular run. Upon each evaluation (five per iteration), 10,000 trials were simulated. Proposed parameter sets were evaluated with quantile maximum likelihood estimation, a discrete approximation to maximum likelihood estimation (Heathcote et al., 2002). Following Walsh et al. (2014), we binned false starts (RTs $<$ 150 ms) separately and binned the remaining RTs according to 20 quantiles. The parameters associated with the best fit were then selected.

Results

Compared to the simplex, the completion times for the PDLT were faster on average (1.32 vs 49.17 sec) and less variable (SD = .02 vs SD = 12.26 sec). We computed the correlation between the time to simulate the model using the recovered parameters and the corresponding completion times for each method. As expected, the correlation was nearly zero for the

PDLT ($r = .04$) but markedly higher for the simplex ($r = .94$). This underscores the inability of the simplex to scale up to slower models requiring more simulation time.

We assessed the ability of each method to accurately recover the parameters with two metrics: relative bias and correlation. Relative bias provides a standardized measure of the deviation between the true and recovered parameters using the following formula: $RB = \frac{(\hat{\theta}-\theta)}{\theta}$, where $\hat{\theta}$ is the estimated parameter and θ is the true parameter. The correlation measures the degree to which the true and recovered parameters are linearly related independent of systematic bias. Lower correlations are indicative of more noise in the parameter estimate. As shown in Table 2, mean relative bias was generally low for both methods, indicating an accurate parameter recovery. In general, the simplex method exhibited slightly less bias than the PDLT method.[1]

Table 2: Mean *RB* between true and recovered parameters.

	Utility	Threshold	FP_{dec}	Cycle Time	Diff
PDLT	.09	.08	.00	.01	-.47
Simplex	.06	.06	.00	.00	-.45

Table 3 shows that the correlations were generally high for both methods, providing more evidence of accurate parameter recovery. In terms of correlation, the PDLT method performed slightly better than the simplex method. Taken together, these results indicate that the PDLT can achieve similar accuracy as the simplex in just a faction of the time. In contrast to the simplex, the independence between simulation time and fit time for the PDLT indicates that it can easily scale up to slower, more complex models.

Table 3: Correlations between true and recovered parameters.

	Utility	Threshold	FP_{dec}	Cycle Time	Diff
PDLT	.89	.74	.64	.89	.93
Simplex	.85	.73	.42	.84	.89

Discussion

Real-time monitoring using cognitive models provides an opportunity to examine changes in cognitive processes, predict performance decrements, and implement adaptive interfaces designed to mitigate performance decrements. A practical challenge in performing real-time monitoring is updating cognitive models with new data as they arrive. Established parameter estimation methods are unable to accurately estimate parameters of simulation-based models in real time. To overcome this limitation, we developed a real-time parameter estimation method, and validated it on a simulation-based ACT-R model of the PVT. PDLT pre-computes the models predictions in order to offload the computational burden associated with simulating model predictions. The results of our simulations demonstrate several important points. First, the PDLT is fast, even with a high sampling resolution and large number of RTs. Second, the PDLT can achieve a similar level of accuracy compared to the simplex method. Third, unlike other methods, the PDLT shows potential to scale up to more complex models due to the separation of simulation time and parameter estimation time. Together, these findings suggest that cognitive models can be updated on a trial-by-trial basis for many tasks.

Scaling Up

Although we used a relatively simple task and model for demonstration, there is reason to believe the PDLT is scalable to more complex models. For example, we demonstrated that the PDLT is invariant to the time required to simulate the model, unlike the simplex method. Other methods that require real-time model simulation for parameter estimation will likely show limited scalability. The ability of the PDLT to scale up can be attributed to the use of pre-computation.

There are some situations in which speed-accuracy trade-offs will be inevitable. For example, complex models that produce multiple responses will require the storage and evaluation of more predictions. In practice, this trade-off may be relatively small given that many tasks permit only a handful of responses. Similarly, dynamic models, whose parameters change during a task, require additional predictions to be stored. We recommend constraining the parameters changes to be some function of time or trial to reduce the parameter space and number of stored predictions. For example, our model could be extended to allow U_S or U_T to change as a function of trial. It may also be advisable to use quantiles rather than kernel density functions for dynamic models in order to reduce computation time. Additionally, some speed-accuracy trade-offs will occur with models that span large parameter spaces and have low correlations among parameters.

Parameter Volatility

A potential challenge with real-time monitoring is dealing with volatility in parameter estimates. One solution might be to incorporate data from previous sessions during real-time monitoring. Incorporating previous data would serve as a sort of quasi-prior, forcing the estimate to stabilize around an informed value. However, this may have the undesirable effect of making fatigue-relevant parameters less sensitive to changes in fatigue. An alternative approach might be to fix parameters that are invariant to changes in fatigue based on previous results. For example, Walsh et al. (2014) found evidence that *cycle time* and FP_{dec} vary across individuals but are invariant to the effects of fatigue. A drawback of this approach is that it would require a separate PDLT for each person. Alternatively, it might be possible to fix certain pa-

[1]The high relative bias for *Diff* is an artifact of its scale. Small values in the denominator of the relative bias formula tend to inflate the resulting value. Absolute bias for *Diff* is attenuated because U_S and U_T are biased in the same direction: $Diff = (U_S + bias_{U_S}) - (U_T + bias_{U_T})$.

rameters across individuals if they have low variance and/or do not contribute substantially to the model fit.

Another method for reducing parameter volatility is constraining the model with physiological data. Walsh et al. (2014) integrated a biomathematical model of fatigue with the ACT-R model, forcing U_S and U_T to vary according to participants' sleep/wake history and circadian rhythm. Physiological constraints serve to stabilize parameter estimates and allow for the detection of meaningful changes.

The quality of an estimate may be improved by leveraging statistical properties of composite parameters. For example, *Diff*—the difference between U_S and U_T—is generally more important than the absolute values of U_S and U_T for understanding and predicting fatigue. From a statistical standpoint, the difference of two random variables has desirable properties. Bias in *Diff* is attenuated because U_S and U_T are biased in the same direction and tend to cancel out each other. In addition, the variance of *Diff* is attenuated because U_S and U_T are correlated. Thus, the volatility of the parameters can be mitigated by exploiting the statistical properties of the model and focusing on key relationships between parameters.

Conclusions

The real-time parameter estimation method that we developed and validated is an important advance toward real-time model-based monitoring. Prior to this effort, real-time monitoring was restricted to performance metrics, such as accuracy and mean RT, and very simple models, such as those used in intelligent tutoring systems (Wilson & Russell, 2003). The PDLT method can be used with any simulation-based model and easily scales to complex models due to the use of pre-computation. We believe this new methodology will enable the use of computational models for real-time monitoring of workload and fatigue and in the implementation of adaptive interfaces. (Rouse, 1988; Green et al., 2009).

Acknowledgments

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This research was supported by a 711$^{\text{th}}$ Human Performance Wing Chief Scientist Seedling grant to G.G. and L.M.B. Distribution A: Approved for public release; distribution unlimited. 88ABW Cleared 06/06/2016; 88ABW-2016-2869.

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2014). Julia: A fresh approach to numerical computation. *arXiv preprint arXiv:1411.1607*.

Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-adapted Interaction*, *4*(4), 253–278.

Dinges, D. F., & Powell, J. W. (1985). Microcomputer analyses of performance on a portable, simple visual RT task during sustained operations. *Behavior Research Methods, Instruments, & Computers*, *17*(6), 652–655.

Gen, M., & Cheng, R. (2000). *Genetic algorithm and engineering optimization*. New York: John Wily and Sons.

Gluck, K. A., & Gunzelmann, G. (2013). Computational process modeling and cognitive stressors: Background and prospects for application in cognitive. *The Oxford handbook of cognitive engineering*, 424.

Green, T. M., Ribarsky, W., & Fisher, B. (2009). Building and applying a human cognition model for visual analytics. *Information Visualization*, *8*(1), 1–13.

Gunzelmann, G., Gross, J. B., Gluck, K. A., & Dinges, D. F. (2009). Sleep deprivation and sustained attention performance: Integrating mathematical and cognitive modeling. *Cognitive Science*, *33*(5), 880–910.

Gunzelmann, G., Veksler, B. Z., Walsh, M. M., & Gluck, K. A. (2015). Understanding and predicting the cognitive effects of sleep loss through simulation. *Translational Issues in Psychological Science*, *1*(1), 106.

Harris, J. (2008). Mindmodeling@home: A large-scale computational cognitive modeling infrastructure. In *Proceedings of the 6th annual conference on systems engineering research.*

Heathcote, A., Brown, S., & Mewhort, D. J. (2002). Quantile maximum likelihood estimation of response time distributions. *Psychonomic Bulletin & Review*, *9*(2), 394–401.

Lim, J., & Dinges, D. F. (2008). Sleep deprivation and vigilant attention. *Annals of the New York Academy of Sciences*, *1129*(1), 305–322.

Loh, S., Lamond, N., Dorrian, J., Roach, G., & Dawson, D. (2004). The validity of psychomotor vigilance tasks of less than 10-minute duration. *Behavior Research Methods*, *36*(2), 339–346.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*(4), 308–313.

Rouse, W. B. (1988). Adaptive aiding for human/computer control. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *30*(4), 431–443.

Van den Bos, A. (2007). *Parameter estimation for scientists and engineers*. John Wiley & Sons.

Veksler, B., & Gunzelmann, G. (Under Review). Functional equivalence of sleep loss and time on task effects in sustained attention.

Walsh, M. M., Gunzelmann, G., & Van Dongen, H. P. (2014). Comparing accounts of psychomotor vigilance impairment due to sleep loss. In *Proceedings of the 36th annual conference of the cognitive science society* (p. 877-882). Pasadena, California.

Wilson, G. F., & Russell, C. A. (2003). Real-time assessment of mental workload using psychophysiological measures and artificial neural networks. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *45*(4), 635–644.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Using a cognitive architecture in educational and recreational games: How to incorporate a model in your App

Niels A. Taatgen (n.a.taatgen@rug.nl) and Harmen de Weerd (h.a.de.weerd@rug.nl)
Institute of Artificial Intelligence, Nijenborgh 9
9747 AG Groningen, The Netherlands

Abstract

We present a Swift re-implementation of the ACT-R cognitive architecture, which can be used to quickly build iOS Apps that incorporate an ACT-R model as a core feature. We discuss how this implementation can be used in an example model, and explore the breadth of possibilities by presenting six Apps resulting from a newly developed course in which students make use of Swift ACT-R to combine cognitive models with mobile applications.

Keywords: ACT-R, mobile apps, game design

Introduction

Cognitive models have proven to be a valuable research tool in advancing our understanding of human cognition. Because of their ability to model human behavior, cognitive models also have a great potential for use outside of research, such as in educational or recreational settings. In this role, the model is not used to explain human data, but to act as a simulated human agent. In previous cases, such as the ACT-R model that played SET (Taatgen et al., 2003), and DISTRACT-R (Salvucci et al., 2005), the model was implemented directly in the target programming language. In this paper, we present an ACT-R re-implementation that can be used as a component in an iOS App. The implementation makes it possible to quickly build Apps with ACT-R inside. As a demonstration, we present a Rock-Paper-Scissors App that the first author built in just one-and-a-half hour. We further look at the results of a course that we taught using the implementation, and the six Apps that came out of that course.

Swift ACT-R

The re-implementation of ACT-R uses the new Swift programming language. Swift is an object-oriented programming language similar to Java and C++. The Swift implementation of ACT-R consists of a set of classes that implement the different components of ACT-R, such as Chunks, Declarative Memory, Procedural Memory, and the overarching Model class.

The simplest way to use Swift ACT-R is to write a text-file with a regular ACT-R model (with some limitations). The next step is to build a controller for the App, that responds to button presses and other actions the user can take. This controller creates an instance of the model class, and loads the ACT-R model into that instance:

```
model = Model()
model.loadModel("example")
```

The model can then be run using the `run` method:

```
model.run()
```

The model communicates with the App through the action buffer (a new buffer that takes the role of standard perception and action buffers). Whenever a production rule takes a `+action>` action, the model stops, and hands control back to the main program. The main program can then read out the contents of the action buffer, make appropriate changes to the interface, wait for user input, place information back into the action buffer, and then run the model again.

There are several alternatives to using ACT-R code, for example, it is also possible to access declarative memory directly, or even to have no explicit ACT-R model, but instead use declarative memory directly. The ACT-R code can be downloaded from:
https://github.com/ntaatgen/ACT-R
It has two example models, both of the prisoner's dilemma (Lebiere, Wallach & West, 2000 and Stevens, Taatgen, & Cnossen, 2016).

Example model: Rock - Paper - Scissors

Lebiere and West (1999) built an ACT-R model that can play Rock-Paper-Scissors, and adapts itself to its opponent by trying to predict the next move based on previous experiences. The lag 1 model of Lebiere et al. stores sequences of two consecutive moves of the opponent in declarative memory and uses these to predict the opponent's next move. For example, the model has the following chunks for sequences that start with rock:

```
(RR isa decision step1 rock step2 rock)
(RP isa decision step1 rock step2 paper)
(RS isa decision step1 rock step2 scissors)
```

Each time the opponent plays rock twice in a row, the RR chunk is strengthened, each time rock is followed by paper, the RP chunk is strengthened, and each time rock is followed by scissors, the RS chunk is strengthened. When the model needs to decide what to do in the turn after the opponent has played rock, it retrieves the most active chunk with rock in step1. The value in step2 is then the model's prediction for the next move of the opponent. It only needs to decide what move to counter that with. The whole model consists of only four production rules (actually, five: one more rule to play the first game, when there is no previous decision). Figure 1 lists these productions.

```
(p retrieve-decision
    =goal>
        isa goal
        state start
        playerlast =last
==>
    =goal>
        state retrieve
    +retrieval>
        isa decision
        step1 =last)

 (p retrieve-beats
    =goal>
        isa goal
        state retrieve
    =retrieval>
        isa decision
        step2 =prediction
==>
    =goal>
        state retrieve-beats
    +retrieval>
        isa beats
        slot1 =prediction)

(p make-decision
   =goal>
     isa goal
     state retrieve-beats
   =retrieval>
     isa beats
     slot2 =decision
==>
   =goal>
     state decide
   +action>
     isa move
     choice =decision)

(p restart-after-action
  =goal>
    isa goal
    state decide
    playerlast =last
  =action>
    isa move
    opponent =decision
==>
  +goal>
     isa goal
     state start
     playerlast =decision
  +imaginal>
    isa decision
    step1 =last
    step2 =decision
  -action>)
```

Figure 1: Productions in the Rock-Paper-Scissors model

The model makes a decision in three steps. It first retrieves its prediction for what the opponent will do next based on their previous move. Based on that prediction, it will retrieve from memory the move that will beat the predicted action (e.g. rock beats scissors). It will then put this action into the action buffer. Control is then returned to the main program, which waits until the human player takes an action by pushing one of three buttons in the interface (Figure 2).

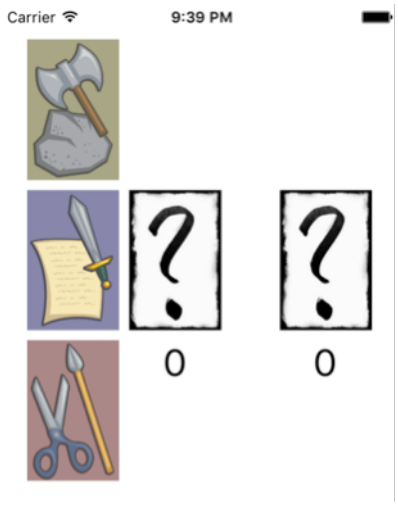


Figure 2. The Rock-Paper-Scissors game on an iPhone

The program itself is straightforward. The function that is called when the App is started already loads in the model and carries out a first run. The model has therefore made its decision, and now waits for the player to tap one of the buttons. Once the player has made a decision, the code checks who has won, and adjust the scores. Figure 3 shows all the basic code that is necessary. Some additional code is needed to update the display with appropriate feedback, and show the scores.

To explore the breadth of possibilities of constructing Apps with a built-in model, we made this the goal of an advanced cognitive modeling course.

Course outline

The course 'Cognitive Modeling: Complex Behaviour' is part of the Master Human-Machine Communication and the Master Artificial Intelligence at the University of Groningen. It has been set up as a so-called *learning community*. For the purpose of the course, students had access to a lab room with several workstations to develop apps on, as well as a number of iPads and iPhones for testing. The lab room was available to the students for the full duration of the ten-week course. In line with the concept of a learning community, the focus is on letting the students present their work for open discussion among themselves, rather than on formal lectures.

The course followed the plan outlined in Table 1. At the first meeting, students divided themselves into three-person

```
 override func viewDidLoad() { // This function is called when the App starts up
    super.viewDidLoad()
    model.loadModel("rps")
    model.run()
}

 // The following function is called when the player pushed one of the buttons
 @IBAction func gameAction(sender: UIButton) {
    // The player action is the title of the button that was pressed
    let playerAction = sender.currentTitle!
    // The model action is in the choice slot in the action buffer
    let modelAction = model.lastAction("choice")!
    // Determine the outcome of the game
    switch (playerAction,modelAction) {
    case ("Rock","rock"),("Paper","paper"),("Scissors","scissors"):
        // Tie
        break
    case ("Rock","scissors"),("Paper","rock"),("Scissors","paper"):
        // Players wins
        pScore += 1
        mScore -= 1
    default:
        // Model wins
        pScore -= 1
        mScore += 1
    }
    // Communicate the player's action back to the model by setting a slot
    // in the action buffer
    model.modifyLastAction("opponent", value: playerAction.lowercaseString)
    // And run the model again for the next trial
    model.run()
 }
```

Figure 3. Code in the App to handle the interaction between the player and the model.

project teams, and were encouraged to immediately start developing a project proposal. A project proposal was subject to two conditions: (1) the App had to be developed in Swift, and (2) the core of the App should be the Swift implementation of the ACT-R cognitive architecture. No further requirements were given, although project proposals had to be approved before a team could start. In particular, students were free to choose to build a game, an educational app, or different applications using an ACT-R model. In addition, students were free to make their App for iPad, iPhone, Apple Watch, or any combination of the three.

Each team consisted of three people, with one member being responsible for graphical user interface (GUI) design, one for cognitive model design, and for programming and coordination. The first two weeks were meant for students to familiarize themselves with the Swift programming language and the Swift ACT-R implementation. Each of these topics included a short lecture and a small, ungraded assignment.

Students presented their finalized project proposals in the third week. Over the following five weeks, students gave weekly progress reports on the status of their project, either privately with one of the lecturers, or as a presentation to fellow students to encourage discussion of common problems and solutions.

Final presentations and demonstrations of the App were due in week 8 and 9, which left the students one additional week to write a final report on their App. Mirroring the structure of the student projects, the final report was required to discuss the graphical user interface, the cognitive model, and general programming.

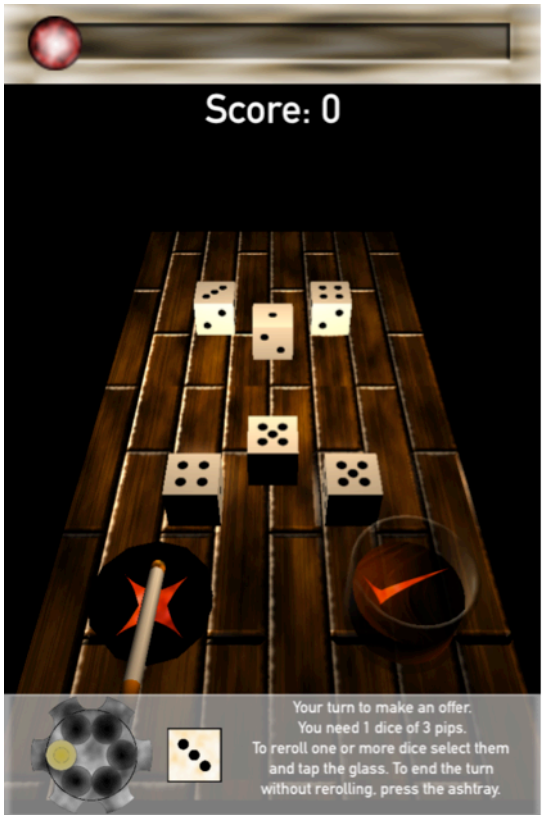
Table 1: Course plan for 'Cognitive Modeling: Complex Behavior'.

Week	Activity
1	Introductory lecture on Swift Creating project teams Assignment: Build simple calculator app
2	Introductory lecture on Swift ACT-R Assignment: Build rock-paper-scissors opponent using Swift ACT-R
3	Presentation final project proposals
4-7	Progress reports
8	Final presentation
9	Demonstration of the App and election of the best App
10	Deadline final report

Description of developed mobile apps

Six projects were developed during the course, each with a corresponding App. As mentioned in the course outline, students were free to choose the topic of their application, as long as it included Swift ACT-R as a core mechanism. The six projects included three recreational games, two educational games, and one other application. In this section, we describe each of these apps in more detail.

Six-Dice game



The six-dice game is a recreational game of incomplete information played over a number of rounds. Two players control three dice each. At the start of each round, each player is given a goal that involves a certain number of dice that should show a given number of pips. For example, the human player in the screenshot above has the goal to have at least one of the six dice show a 3, while the cognitive agent may have the goal to have one die to have rolled a 6. Note that these goals are private information. That is, neither player knows the other player's goal.

Once the goals are revealed, all dice are rolled and revealed. Next, one of the players may offer to reroll any subset of their own three dice. The other player must decide whether or not to accept this proposal. If the proposal is rejected, the round ends and each player who has achieved his or her goal gains one point. If the second player accepts the proposal, the dice selected by the proposing player are rerolled, but the second player also has to select the same number of their own dice to reroll. Note that the deciding player controls which dice are rerolled.

At the end of the game, the player with the highest score wins. However, when the combined score of both players is below a certain threshold, the game ends without a winner. The game is therefore a game of mixed motives. Especially near the end of the game, it may be in the best interest of a player to allow the opponent to reach their goal.

The ACT-R model is used to assess the opponent's trustworthiness. Each game, the model would assess the outcome of the game, and assign it a trust value between 1 and 10, and add this as a chunk to declarative memory. When it later had to make a decision in which trust of the other player played a role, it would perform a blended retrieval of the trust value.

Memory

Memory (also known as Concentration) is a recreational card game played by placing a number of cards face down on a surface. Players take turns revealing two of the cards. If these two cards form a pair, the cards are removed from the game and the player scores a point. Otherwise, the cards are placed back face down. The game continues until no cards are left, at which point the player with the most points wins.

Note that for a computer player, the game of Memory is a trivial one. After all, turning the cards face down after revealing only presents a challenge for players without a perfect memory. The goal of incorporating a cognitive agent in Memory is therefore to create a fun and challenging competitor, rather than to make an agent that follows an optimal strategy in playing Memory.

The ACT-R Memory player makes use of declarative memory to store card information such as the identities and positions of previously revealed cards. When the ACT-R player believes to have found a pair of cards, it tries to retrieve the locations and claim the pair. To simulate human-like errors, the model adds noise to the stored positions of cards. This causes cards on the edges and corners of the surface to be remembered better than interior cards.

Pyramid game

The pyramid game is a recreational game played with a standard deck of 52 cards. Ten cards are placed face down as a pyramid (see screenshot above). In addition, each player receives four facedown cards. At the start of the game, each player is allowed to look at their own four cards. A player cannot look at the cards of the other player, and once the game starts, a player is no longer allowed to review their own cards either.

The game is divided into multiple rounds. In each round, a face down pyramid card is turned over, starting at the bottom left and moving slowly up the pyramid. The value of a card depends on its position in the pyramid. Cards on the bottom row are worth 1 point, while the top card is worth 4 points. Once a card is revealed, both players decide whether or not to claim that one of their four cards has the same face value. If a player decides to do so, they select one of their four cards. The other player may then choose to challenge this claim by turning over the selected card and check its face value.

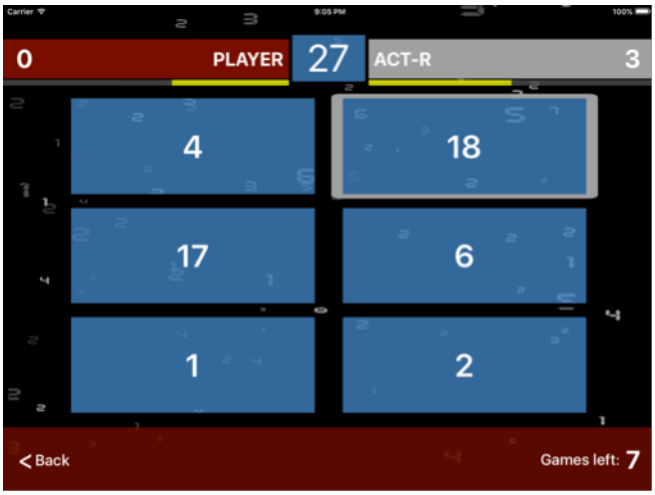
The players' scores change based on the outcome of a round. If no claim is made, scores remain unchanged. When a claim remains unchallenged, the claimant adds the value of the pyramid card to their score. If the claim is challenged and found to be false, the challenger adds twice the value of the pyramid card to their score. Finally, if a claim is challenged and found to be true, the claimant adds twice the value of the pyramid card to his score.

Independent of whether a claim was challenged, any card that was used to claim points are replaced with new cards from the deck. Only the owner of a new card is allowed to inspect it.

For the purpose of the app, the human player always starts by deciding to make a claim. Once this claim is resolved, the computer player takes its turn and the game continues to the next pyramid card. When there are no cards left on the pyramid to turn over, the player with the highest score wins.

The ACT-R opponent makes use of declarative memory to remember its own cards, and regularly rehearses these cards to avoid forgetting. In addition, the ACT-R player tries to model the behavior of the human player in terms of the likelihood that the human player is bluffing and the likelihood that the human player will challenge a claim of the ACT-R player.

Mathgician game



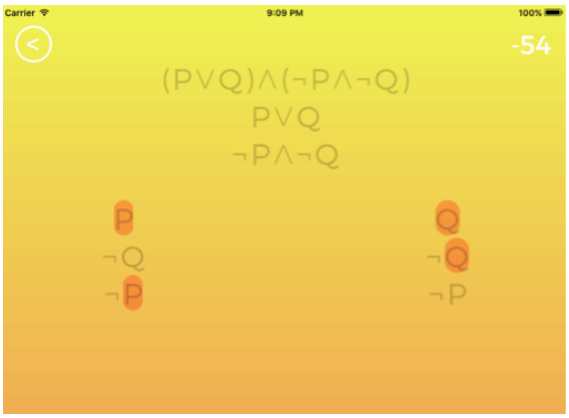
Mathgician is an educational App aimed at training addition to children in the form of a competitive game. In the game, players are presented with a goal number and six tiles. Each tile has a number printed on it. Players have to select tiles such that the numbers on the selected tiles add up to the goal number. For example, to get the goal of 27, as in the screenshot above, players could select the tiles with the numbers 17, 6, and 4, but also the tiles with the numbers 18, 6, 2, and 1.

The game is played against an ACT-R opponent, which follows a human-like greedy strategy, in which it tries to get to the goal number with high numbers first. If this fails, the model tries lower numbers. In addition, the App has the option to play against an adaptive opponent, which tries to match the search speed of the ACT-R model with the behavioral data of the human player.

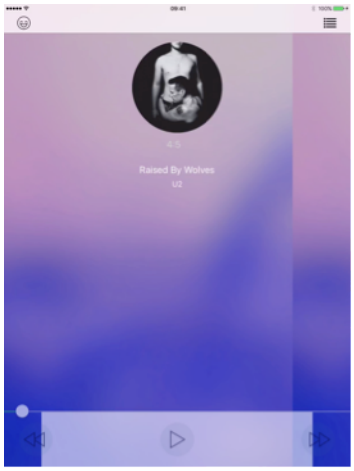
OMGLogic game

OMGLogic is an educational App that is intended to help players learn how to construct semantic tableaux. In the app, players are presented with formulas from propositional logic. They are asked to construct a semantic tableau to show that the formula cannot be satisfied. To do so, players have to select part of the formula and decide how it should be handled to continue the tableau. Whenever a player takes a correct action their score increases, while incorrect answers decrease a player's score.

The ACT-R model presents a competitor that attempts to gain points for itself while constructing the semantic tableau. The ACT-R model attempts to match the skill level of the human player in solving the formulas. If the model successfully retrieves a correct course of action before the human player, the step is executed and the human player loses points.



LagMusic app



LagMusic is a music player that makes use of ACT-R to predict when a listener wants to listen to a previously heard song again, given a user-specified mood. The model uses the actions of the user as feedback. Skipping a song is considered to be an indication that the user is unwilling to listen to the song, while a user that listens to a song for the full duration is considered to be happy with the model's selection. Finally, a user can indicate that it likes a song, but does not want to listen to it at the current moment by shaking the device.

The ACT-R model uses activation of a song chunk to determine whether, given the user's current mood, a song has sufficiently faded from memory for the user to appreciate hearing it again, as well as whether or not the user appreciates the song at all in the current mood. Whenever a song is played in full, the positive feedback increases activation. Skipping a song provides negative feedback by decreasing a song's activity. In addition, the model updates the retrieval threshold, making it more likely for the model to retrieve songs with higher activation. The neutral feedback, which indicates that the song is appreciated but played too soon, adjusts the retrieval threshold in the other direction, making it more likely for the model to retrieve songs with lower activation.

Conclusion

Cognitive models are not only useful to build theories of human cognition, but they can also be applied to build simulated humans. In this paper we explored possible ways in which the ACT-R architecture can be used in the context of a mobile application. One of the conclusions we can draw at this stage is that for most purposes declarative memory is the most useful component in ACT-R. It can model how we remember and forget information, and can also model decision making through instances-based learning. Further potential of incorporating a model in an App is that the App can gather its own information to train the model. Again, declarative learning is the lowest hanging fruit here, but procedural learning is potentially very powerful as well.

Acknowledgments

The underlying research project is funded by the Metalogue project; a Seventh Framework Programme collaboration funded by the European Commission, grant agreement number: 611073.

We want to thank the following students for their efforts in bringing the ACT-R Apps to life: Harmke Alkemade, Roberto De Cecilio De Carlos, Andrija Curganov, Thomas Derksen, Annemarie Galetzka, Joram Koiter, Michael LeKander, Milena Mandic, Rick van der Mark, Hugo van Plateringen, Tom Renkema, Jordi Top, Teun van Tuijl, Olaf Visker, Alex de Vries, Evert van der Weit, Marco Wirthlin, and Maikel Withagen.

References

Anderson, J.R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford, USA: Oxford University Press.

Lebiere, C., Wallach, D., & West, R. (2000). A memory-based account of the prisoner's dilemma and other 2x2 games. In N. Taatgen, & J. Aasman (Eds.), *Proceedings of the Third International Conference on Cognitive Modeling* (pp. 185–193). Veenendaal: Universal Press.

Lebiere, C., & West, R.L. (1999). Using ACT-R to model the dynamic properties of simple games. In *Proceedings of the Twenty-first Conference of the Cognitive Science Society* (pp. 296-301).

Salvucci, D.D., Zuber, M., Beregovaia, E., & Markley, D. (2005). Distract-R: Rapid prototyping and evaluation of in-vehicle interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 581-589).

Stevens, C.A., Taatgen, N.A., & Cnossen, F. (2016). Instance-based models of metacognition in the prisoner's dilemma. *Topics in Cognitive Science, 8(1)*, 322-334.

Taatgen, N.A., van Oploo, M., Braaksma, J. & Niemantsverdriet, J. (2003). How to construct a believable opponent using cognitive modeling in the game of Set. In *Proceedings of the Fifth International Conference on Cognitive Modeling* (pp. 201-206).

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

JIVUI: JavaScript Interface for Visualization of User Interaction

Ignacio X. Domínguez (ignacioxd@ncsu.edu), **Jayant Dhawan (jdhawan2@ncsu.edu)**,
Robert St. Amant (stamant@csc.ncsu.edu), **David L. Roberts (robertsd@csc.ncsu.edu)**
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206 USA

Abstract

In this paper we describe the *JavaScript Interface for Visualization of User Interaction* (JIVUI): a modular, Web-based, and customizable visualization tool that shows an animation of the trace of a user interaction with a graphical interface, or of predictions made by cognitive models of user interaction. Any combination of gaze, mouse, and keyboard data can be reproduced within a user-provided interface. Although customizable, the tool includes a series of plug-ins to support common visualization tasks, including a timeline of input device events and perceptual and cognitive operators based on the Model Human Processor and TYPIST. We talk about our use of this tool to support hypothesis generation, assumption validation, and to guide our modeling efforts.

Keywords: Typing; Cognitive Modeling; Visualization; Replay; Tool; Data Visualization; Input Device; Cognition; Inter-keystroke Interval; Mouse Clicks; Mouse Motion; Eye-tracking; Gaze Data; Human-Computer Interaction; Human Information Processing; Human Behavior.

Introduction

Visually representing data is a common technique used to a) succinctly summarize and communicate information, and b) gain a better understanding of the process that generated said data. In its most common form—charts—data visualization is generally used for the former. For the latter, however, visualizations are generally more complex and are built ad-hoc according to the originating data's context. This results in an increased effort required to visualize similar, but not identical datasets.

Examples of data that are generally visualized to be better understood come from cognitive modeling and empirical user interaction with graphical user interfaces. In both cases, the data correspond to a particular context (*e.g.*, the user interface being used or modelled), to a particular set of input devices (*e.g.*, mice, keyboards, eye-trackers, trackpads, etc.), and has a particular set of spatial and temporal properties (*e.g.*, locations on the screen or relative to on-screen items, time elapsed between keyboard or mouse events, etc). A consequence of this need for specialization is a reduced number of available tools that are suitable to visualize a specific dataset. In cases where there exists a suitable tool, an added inconvenience is that it may not support the researcher's preferred operating system, or it may require the installation of dependencies that the researcher may prefer not to install.

In this paper we present the JavaScript Interface for Visualization of User Interaction (JIVUI)—a modular, Web-based, and customizable tool that addresses these problems, providing the ability to visualize and replay a user's interaction with a graphical user interface on any platform through any modern Web browser.

Data produced from cognitive models of input device usage is generally similar to that collected from a user, with the addition of annotations for the cognitive operators related to the user's observed interactions. Through a flexible data representation, JIVUI supports both. In addition to user and/or model data, JIVUI is capable of rendering a replay of the interaction over a customizable UI that can be crafted to represent the task from which the data was collected.

Related and Prior Work

Visualization tools are a staple in cognitive modeling research. Models can grow to be very complex, and understanding the subtleties of their behavior in carrying out specific tasks often requires more than textual descriptions or traces, tables of quantitative trace information, or summary statistics. A variety of tools are available for presenting models and user data in graphical form.

The ACT-R 6.0 Environment (Bothell, 2004) provides controls and information about a model in a graphical user interface. A stepper function is described as the most useful tool in the environment. It is comparable to a stepper in a conventional programming environment, supporting pauses before events and a range of choices for running a model until a specified condition becomes true (when a given duration is exceeded, a given production fires, or an event is generated by a given module). A graphical trace is provided, with time along one axis and events of different types in rows, filling the other axis. The environment can also display a state chart diagram, a directed graph showing the productions selected and fired in a model. Other, more specialized visualizations are also supported.

SANLab-CM (Patton & Gray, 2010) is a tool for activity network modeling, specifically models that include stochastic operators. SANLab-CM gives a modeler the ability to construct and edit a model in the form of a specialized directed graph that captures elementary cognitive, perceptual, and motor operators and the dependencies between them, as well as compositions called interactive routines. The modeler can visualize the model as a Gantt chart that shows the execution of operators over time; a histogram shows the distribution of model execution times. By selecting a specific critical path, the modeler can focus the visualization on the associated subset of operators.

NAV (Kriete, House, Bodenheimer, & Noelle, 2005) is a tool for generating animations of cognitive model execution, showing different visualizations to help non-expert users understand the dynamics of a model. The focus is on giving a

Figure 1: Example of replay and control interfaces illustrating the playback of typing and gaze data.

modeler the facilities needed to build a presentation-quality animation to convey the necessary information. For example, in a semantic network visually represented as nodes and arcs, the modeler can opt to show node activation levels over time by changing color or size. Annotations can be added to the visualization, which can also change over time.

SIMCog-JS (Simplified Interfacing for Modeling Cognition - JavaScript) (Halverson, Reynolds, & Blaha, 2015) addresses the continuing challenge of building cognitive models that can interact with external software, in this case the interaction between Java ACT-R and HTML/JavaScript. SIMCog-JS does not provide visualization facilities per se. Due to its integration with JavaScript, however, SIMCog-JS makes it very easy to instrument a Web interface to replay the behavior of a user interacting with the interface or to simulate a cognitive model using the interface—*e.g.* Dong and St. Amant (2016). One novel use of SIMCog-JS in this way has been toward the visualization of eye movements (Balint, Reynolds, Blaha, & Halverson, 2015). A Model Visualization Dashboard can play the trace of a model performing tasks in a given interface, along with other types of visualizations. Further, the environment contains an embodied virtual character that simulates eye and head movements predicted for a human being carrying out tasks in the interface.

An Overview of JIVUI

JIVUI is a Web framework designed to simplify the visualization of gaze, keyboard, and mouse data, and can visually represent perceptual and cognitive operators associated with the data on a timeline. It supports data generated by a cognitive model as well as data collected empirically. The data is provided to JIVUI in JSON (ECMA International, 2013) format either in the page's source or by loading it through the Web interface. The JSON structure and data flow are explained in more detail in the **JSON Data Description** section.

We designed JIVUI to run completely in a Web browser without the need for a back-end, making it immediately usable on any platform. A typical JIVUI instance will render a Web page with a replay interface, a playback control interface, and a timeline. The replay interface, illustrated at the top of Figure 1, renders the animated user interaction over a user interface that can be customized to look and behave like the experimental environment seen by participants. The playback control interface, illustrated at the bottom of Figure 1, is used to start, pause, and stop the replay animation, as well as to specify the playback speed and to move forward and backward in the animation. Finally, the timeline, shown in Figure 2, will display interaction events such as keystrokes, clicks, and gaze fixations, but can also display perceptual and cognitive operators associated with the interaction events based on a cognitive model.

JIVUI is designed to be extensible, meaning that all of the components described above are provided as plug-ins that can be modified or replaced without modifying JIVUI's core. This plug-in-based architecture allows the visualization experience to be highly customizable to support a wide range of user interfaces and data attributes. JIVUI's extensibility is made possible through its support of plug-ins for almost all its functionality, as described in the **JIVUI's Architecture** section.

JSON Data Description

JIVUI works with millisecond precision. It expects a JSON string containing a "`settings`" object and a "`data`" object (an example is shown in Listing 1). The "`settings`" object is required to contain a "`start`" numeric attribute indicating the smallest millisecond contained in the data, and an "`end`" numeric attribute indicating the largest millisecond contained in the data. It can also contain three optional attributes, "`title`", "`startOffset`" and "`endOffset`", providing a label to the data, time padding at the beginning, and time padding at the end of the visualization, respectively.

The "`data`" object is where user or model data will be provided. It is expected to contain an arbitrary number of entries where the keys are milliseconds. Each entry can contain any combination of event types that occurred at that millisecond, keyed by the event type as follows: "`click`" for mouse clicks, "`key`" for keystrokes, "`mouse`" for mouse position, and "`gaze`" for eye-tracking data. In turn, each of these event types will include a set of required and optional attributes. Because multiple clicks can occur at the same time (*e.g.* from different mouse buttons), a "`click`" event is an array where each element is expected to contain the `x` and `y` coordinates where the click happened, as well as the button that was pressed (*e.g.* "left", "right", or "middle"), and it can also contain an optional "`duration`" attribute indicating for how long was the mouse button pressed. If not specified, a mouse click is assumed to last 100ms for visualization purposes. Multiple keystrokes can also occur at the same time, so a "`key`" event is also an array where each element is expected to contain the key that was pressed, and can also contain optional attributes for the "`duration`" of the keystroke, whether

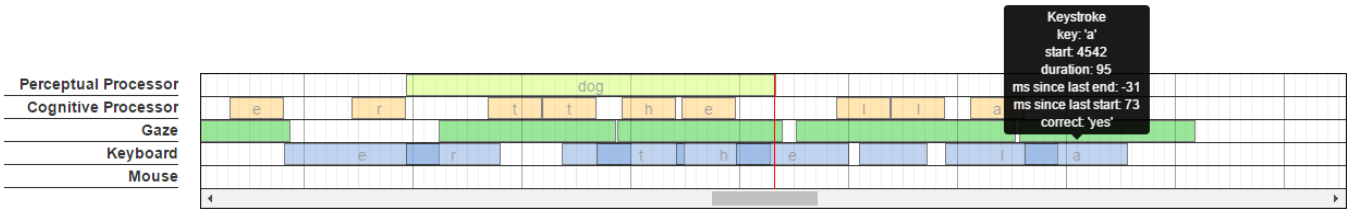


Figure 2: Timeline

```
{
  "settings": {
    "title": "Participant 31", // optional
    "start": 200,
    "end": 250,
    "startOffset": 100,        // optional
    "endOffset": 200           // optional
  },
  "data": {
    "200": {
      "click": [{
        "button": "left",
        "x": 200, "y": 150,
        "duration": 60         // optional
      }],
      "key": [{
        "key": "t",
        "duration": 88,        // optional
        "correct": true,       // optional
        "word": "the"          // optional
      }]
    },
    "250": {
      "gaze": {
        "x": 220, "y": 170,
        "fixated": false       // calculated
      },
      "mouse": {
        "x": 210, "y": 151,
        "drag": true           // calculated
      }
    }
  }
}
```

Listing 1: Example of simple input data in JSON format.

the keystroke correctly matched an expected keystroke (useful for transcription typing, or when the expected keystroke is otherwise known), and a word context where the keystroke occurred. If not specified, a keystroke is assumed to last 100ms for visualization purposes. The "`gaze`" and "`mouse`" events are expected to contain the `x` and `y` screen coordinates of the position of the eye-tracking and mouse pointer data, respectively.

Following the JSON standard, this data representation provides a few advantages. First, it guarantees that there is only one data entry per millisecond. Second, it guarantees that every entry contains at most one instance of each event type. Third, attributes and objects can be added to the data either before it is input to JIVUI, or dynamically by plug-ins, as discussed in the **PLUG-INS** section. More importantly, this representation is a string that can be easily produced from empirical data as well as from the output of cognitive modeling tools, and is supported by virtually all programming languages either natively or through popular libraries.

JIVUI's Architecture

At its core, JIVUI consists of a timing engine, a state manager, and a plug-in manager. The timing engine is used on playback to maintain a stable animation framerate regardless of any browser's constraints or performance limitations. Because JIVUI works with millisecond precision, the default framerate is 1000 frames per second (FPS) as an attempt to render the visualization in real-time. This value is configurable, even as an animation is being played. When the timing engine cannot render at the specified speed, it provides the number of frames that are lagging behind. This allows JIVUI to compensate for this lost time to ensure that the animation has accurate timing based on the set speed and the data's total duration.

As the name suggests, the state manager keeps track of JIVUI's state, which includes the currently loaded dataset (if any), the playback speed, the current frame, and whether the animation is playing, paused, or stopped. The state manager also provides an API to control an animation. Specifically, through the state manager a module can control the animation speed, advance and rewind to a specific frame, and can play, pause, and stop the animation.

The plug-in manager maintains a list of registered plug-ins, and provides an API to register and remove them.

Plug-ins

JIVUI supports two types of plug-ins: preprocessors and UI modules. We designed the plug-in API so that a single component could serve as a preprocessor and a UI module in the interest of maximizing JIVUI's applicability to varied use cases. Each plug-in is described by a JSON file that lists all the resources that it depends on, such as JavaScript files, HTML templates, and/or CSS files.

Preprocessors are exclusively invoked on data load and their main task is to augment the data by performing calculations on it. For example, a preprocessor can be used on gaze data to determine fixations, or work with mouse motion and click data to add attributes that indicate when the mouse motion is occurring within the context of a drag operation.

As shown in Figure 3, JIVUI invokes every preprocessor by calling their "`onDataLoaded(settings, data)`" method,

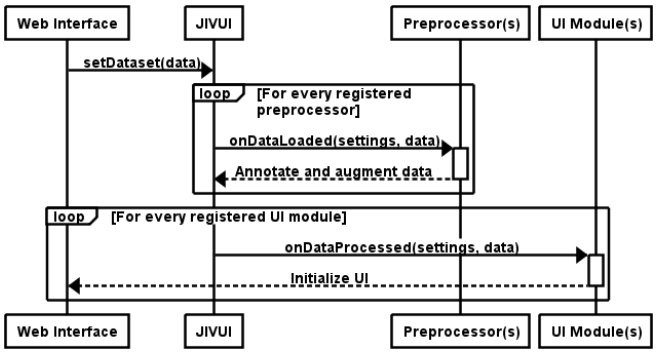


Figure 3: Sequence diagram of the data loading phase, where preprocessors annotate and augment the data, and UI modules initialize the Web interface.

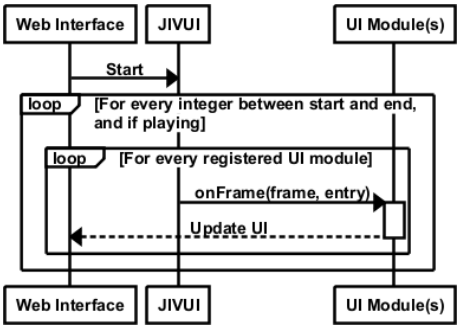


Figure 4: Sequence diagram of the playback phase, where UI modules are provided frame and event data to be rendered.

providing the dataset's "`settings`" and "`data`" objects as arguments. This tells every preprocessor that a new dataset has been loaded and allows them augment data entries.

UI modules are responsible for rendering elements on the Web page and providing functionality to the end user. They are initialized on data load, and are invoked by JIVUI during playback, following the process illustrated in Figure 4. Examples of UI modules are a replay area to visually reproduce keystrokes and mouse motion, playback controls, and timelines.

JIVUI's Included Plug-ins

In order to make JIVUI usable out of the box, and to showcase its potential, we include with it a set of plug-ins that are suitable for the most common user interaction cases. We include basic preprocessors for gaze, keyboard, and mouse data, as well as a cognitive preprocessor. For UI modules, we include a replay region, a set of animation controls, and a timeline.

Preprocessor Modules

Gaze Preprocessors: Two gaze preprocessors are included. One determines fixations using an implementation of the I-DT algorithm as described by Salvucci and Goldberg (2000). For every gaze event, it adds a boolean attribute "`fixated`". The second one uses an implementation of the 1€ filter (Casiez, Roussel, & Vogel, 2012) to stabilize noisy gaze data, adding numeric "`filteredX`" and "`filteredY`" attributes with smoothed eye position values.

Mouse Preprocessor: The included mouse preprocessor augments mouse motion data by adding a boolean "`dragged`" attribute to every mouse motion event. When a mouse motion event occurs within the duration window of a click, as specified by the click's "`duration`" attribute, this value is set to true; otherwise it is set to false.

Keyboard Preprocessor: The keyboard preprocessor augments keystroke events by adding the number of milliseconds elapsed between the start of the current keystroke and the end of the previous one, as well as between the start of the current keystroke and the start of the previous one.

Cognitive Preprocessor: This preprocessor annotates keystroke data with cognitive and perceptual operators based on the Model Human Processor (MHP) (Card, Moran, & Newell, 1986) and the TYPIST model (John, 1996). For every keystroke, this preprocessor creates a cognitive operator that precedes it, lasting 50ms. Similarly, before the first cognitive operator of every word, this preprocessor includes another cognitive operator that also lasts 50ms. Also for every word, a 340ms perceptual operator is created. The TYPIST model describes a working memory capacity of three words, thus perceptual operators for three words are created by this preprocessor at the beginning of the timeline; it waits for the typing of the first word to be completed before creating the perceptual operator for the next word, and so on.

UI Modules

Replay Region: This UI module instantiates a text area where keystrokes appear as key events are received. It also loads two additional UI modules: one to overlay gaze data and the other to overlay mouse motion and clicks. The gaze and mouse UI modules consist of a "`canvas`" element each, and render gaze and mouse motion as moving dots overlaid on the interface. Click events appear as two concentric circles. To improve visibility, the eye and mouse indicators are rendered in different colors that contrast with the background, and are configured to leave a "trail" that fades away as the animation progresses.

Animation Controls: This UI module is used to manipulate the animation. It provides controls to move the animation forward or backward to any particular frame, buttons to start, stop and pause the animation, and controls to set the animation's speed. As the animation is played, this plug-in updates the current millisecond being visualized to reflect the current frame in the animation. It also provides two keyboard shortcuts to control the animation: pressing the space bar will play and pause the animation, while pressing the backspace key will stop it.

Timeline: The included timeline plug-in is designed to look for the augmented data provided by the preprocessors described above and displays events and operators in five different tracks: gaze, mouse, keyboard, cognitive, and perceptual, as shown in Figure 2. Events on each track are displayed as boxes with their length based on the event's duration, and

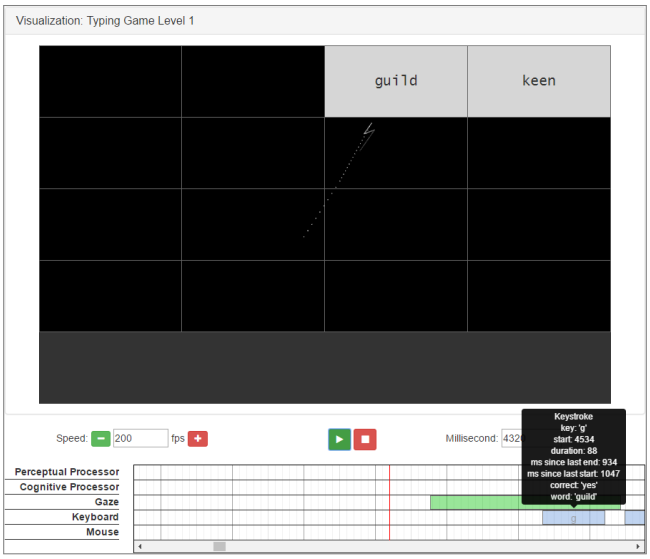


Figure 5: Example of JIVUI being used to visualize gaze and keystroke data collected from *The Typing Game*.

can sometimes overlap. The gaze track displays fixations, the mouse track displays clicks, and the keyboard track displays keystrokes. The cognitive and perceptual tracks display their corresponding operators. Each element on the timeline displays a tooltip on mouse hover providing more information about the event, such as the millisecond value when it occurred, its duration, and other corresponding data (*e.g.*, position of a click, key pressed on a keystroke, etc.), as well as annotations provided by preprocessors, such as the time elapsed after the previous event of the same type.

JIVUI in Use

We used JIVUI to visualize the data we collected from a study involving a computer game, which consisted of participants' typing and gaze data. We were able to replay players' experiences from trace data as a simulation of their participation. A screenshot of one of the play traces is shown in Figure 5. As our study only collected participants' typing and gaze data, we used every plug-in included with JIVUI with the exception of the mouse preprocessor. In this case, we used a modified version of the replay region to mimic the game's user interface, and to respond appropriately to the simulated key presses as if it were the real game.

The use of JIVUI facilitated hypothesis generation for our study. The visualization helped us quickly identify touch typists among our participants, and to generally assess their typing skill, by looking at how often a participant's gaze suddenly dropped off the bottom of the screen, indicating that the participant was looking at the keyboard while typing.

One slightly unexpected discovery in our use of JIVUI came in the examination of typing patterns in the typing game. Participants were asked to type the sentence, "The quick brown fox jumps over the lazy dog," so that the game could be adjusted to their approximate typing speed. Our initial assumption was that this sentence would be typed in a similar way to transcription typing. Following the TYPIST model of transcription typing, we would expect a visual attention shift to a word to occur several keystrokes before the first letter in the word. Possibly because of the way the interface is set up, with the text to be typed visible on the screen, overwritten by gray characters as each is correctly typed, a different pattern typically emerged: the visualization showed that their gaze tended to stay on a word until the word was almost completely typed. We judge that without the animated replay provided by the visualization, it would have taken much longer for us to realize that one of our basic assumptions (the appropriateness of the model) was incorrect.

As a result, we have a new ACT-R model (still under development) with tighter constraints between perceptual and motor processing. In the model, a visual attention shift (including eye movements, using the EMMA (Salvucci, 2000) extension to ACT-R) occurs at the end of each word. The model under-predicts the elapsed time between the fixation on a word and the first keystroke for that word, at around 150 ms, where the elapsed time for participants is closer to 400 ms. Despite this, the model does preserve a general pattern, a relatively small variance in the distribution of elapsed times between fixations and the first keystrokes of words. Our modeling effort in this area continues.

JIVUI also allowed us to assess the quality of our gaze data, helping us identify eye tracker calibration biases, and poor gaze data in general. Being able to visualize calibration biases is incredibly helpful as a guide to correct a participant's gaze data, making it usable, whereas it would have otherwise been discarded.

As a Web tool, we deployed JIVUI on a centralized server allowing multiple researchers to use it simultaneously and on the same dataset. Some of our ongoing efforts using JIVUI to visualize our game data include identifying word segmentation. One approach for doing so is to allow multiple people to visualize our data and manually label segmentation boundaries, resulting in a dataset from which agreement metrics could be computed to determine word segments.

To showcase JIVUI's versatility, we also show in Figure 6 a visualization of data we collected from another game. In this case we visualize mouse motion (overlaid on the game screen) and mouse clicks (as timeline events.

Discussion and Conclusion

The main contribution of this paper is JIVUI itself, which is being made available online[1]. In addition to the source code, we are also providing basic documentation, example data files, and a working demo.

The most salient advantages of JIVUI are its flexible data format, its extensibility, and its portability. With minimal expectations on data structure, JIVUI can be used with both simple and complex datasets, where extended attributes could be easily added to configure the visualization (through the

[1] http://go.ncsu.edu/jivui

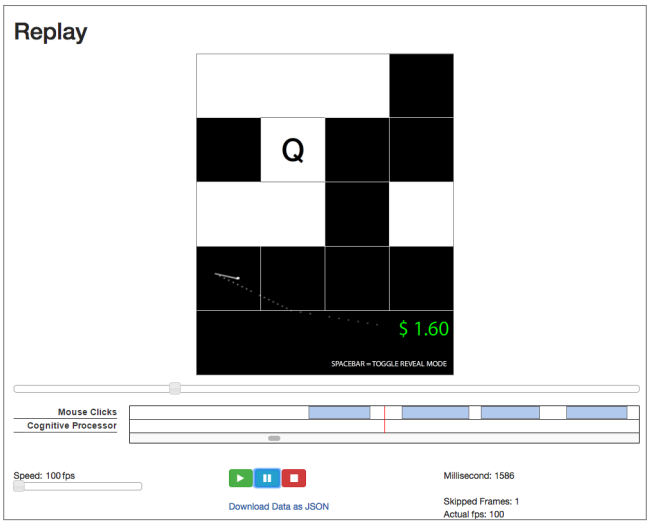


Figure 6: Example of JIVUI being used to visualize mouse motion and click data collected from *The Concentration Game*.

"`settings`" object), and to provide a richer dataset (by adding elements to the data entries). This allows JIVUI to support input data generated from different cognitive modeling methods, as well as data collected from a real user.

JIVUI's heavy reliance on plug-ins allows it to be highly customizable not just to support a wide range of data attributes to visualize, but also to perform calculations on the data and to completely customize how the data is finally played back in the Web interface. Additionally, JIVUI's included plug-ins are designed to accommodate the most common user interaction visualization use cases, making it approachable for newcomers. The included plug-ins also serve as examples for end users to build upon when creating more complex preprocessors and UI modules.

Because JIVUI is based entirely on Web standards, it requires only a modern Web browser to run. This allows JIVUI to be platform-agnostic. It can be hosted on a local computer or on a server, allowing a single instance to be used by multiple people simultaneously. JIVUI does not require any specific dependencies and can be hosted on any Web server. With appropriate UI modules, JIVUI can be used to visualize data on mobile devices as well.

In addition to presenting the tool itself, we have also provided a few examples of how JIVUI has helped us visualize and obtain insight into data collected from our experiments. It can also be applied to visually compare predicted user behavior as cognitive models with actual user data.

Acknowledgments

This work was funded in part by the National Security Agency under grant number H98230-14-C-0139 and in part by the National Science Foundation under grant number IIS-1451172.

References

Balint, J. T., Reynolds, B., Blaha, L. M., & Halverson, T. (2015). Visualizing eye movements in formal cognitive models. In *Workshop on eye tracking and visualization.*

Bothell, D. (2004). Act-r environment manual (working draft) [Computer software manual]. (Downloaded March 2016.)

Card, S. K., Moran, T. P., & Newell, A. (1986). The model human processor: An engineering model of human performance. *Handbook of Perception and Human Performance*, *2*, 1–35.

Casiez, G., Roussel, N., & Vogel, D. (2012). 1€ filter: A simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the 2012 acm conference on human factors in computing systems (chi 2012)* (pp. 2527–2530).

Dong, L., & St. Amant, R. (2016). Answering questions with line and bar graphs. In *International conference on social computing, behavioral-cultural modeling, & prediction and behavior representation in modeling and simulation.* (To appear.)

ECMA International. (2013). *Standard ECMA–404: The JSON Data Interchange Format* (1st ed.).

Halverson, T., Reynolds, B., & Blaha, L. (2015). Simcog-js: Simplified interfacing for modeling cognition - javascript. In *Proceedings of the international conference on cognitive modeling* (pp. 39–44).

John, B. E. (1996). Typist: A theory of performance in skilled typing. *Human-Computer Interaction*, *11*(4), 321–355.

Kriete, T., House, M., Bodenheimer, B., & Noelle, D. C. (2005). Nav: A tool for producing presentation quality animations of graphical cognitive model dynamics. *Behavior research methods*, *37*(2), 335–339.

Patton, E. W., & Gray, W. D. (2010). Sanlab-cm: A tool for incorporating stochastic operations into activity network modeling. *Behavior Research Methods*, *42*(3), 877–883.

Salvucci, D. D. (2000). A model of eye movements and visual attention. In *Proceedings of the 2000 international conference on cognitive modeling* (pp. 252–259).

Salvucci, D. D., & Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 symposium on eye tracking research & applications* (pp. 71–78). doi: 10.1145/355017.355028

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Explaining Mistakes in Single Digit Multiplication: A Cognitive Model

Trudy Buwalda[1] (t.a.buwalda@rug.nl), Jelmer Borst[1] (j.p.borst@rug.nl), Han van der Maas[2], and Niels Taatgen[1] (n.a.taatgen@rug.nl)
[1] University of Groningen, Groningen, the Netherlands
[2] University of Amsterdam, Amsterdam, the Netherlands

Abstract

Error patterns for arithmetic problems are very rich in information, but they are hard to investigate systematically because of the small number of mistakes made. To be able to investigate errors in arithmetic we therefore used an online educational application called Math Garden, which teaches children arithmetic in the form of several different tasks. Because of the large number of users, Math Garden provides sufficient data to investigate errors systematically. Using the Math Garden data set, we developed a cognitive model in the PRIMs architecture that can give a comprehensible account of the errors made in single-digit multiplication problems. The model does a relatively good job of explaining errors on easy problems, but has difficulties explaining mistakes for harder problems. In addition to the current model, we propose some approaches to improve the model to explain mistakes in the harder problems as well.

Keywords: Arithmetic, Multiplication, PRIMs, Errors

Introduction

Arithmetic is one of the core skills taught in primary education. Especially single digit multiplication is considered to be one of the core abilities in today's world. This kind of basic skill is hard to study in adults, because it is so well trained and automatic that hardly any mistakes are made. Even children make few errors. This is unfortunate, because errors give great insight in the processes and especially the strategies underlying these skills.

In this paper, we will investigate the different kinds of errors children make in multiplication by employing a huge data set from a web-based practice program, Math Garden (e.g. Klinkenberg, Straatemeier, & Van Der Maas, 2011). Math Garden is used by thousands of students every day, and the data therefore include a significant amount of errors. To explain the errors children make, we will develop a comprehensible cognitive model of these error data. Our goal is to gain insight into the processes and strategies underlying single-digit multiplication.

Existing Models of Multiplication

Previously, several models have been built to explain patterns found in arithmetic data. We can discern three categories in this regard: memory strength models, network interference models, and computational efficiency models (Ashcraft & Guillaume, 2009). Memory strength models and network interference models both put the emphasis on memory retrievals, while strategy-based models implement the use of algorithmic strategies, such as repeated addition – e.g. solving `8 + 8 + 8` instead of `3 x 8` – and counting in steps of two, three, or more – e.g. `3, 6, 9` to solve `3 x 3`. Most models combine a strategy-based approach with a memory-based approach: retrieval based models often include some kind of computational strategy, and strategy based models often also include a rehearsal strategy (e.g. Lebiere, 1999; Siegler, 1988).

An example of a combined model is the model by Siegler (1988). The main strategy in this model was retrieval, which was tried multiple times. Every trial a random number of retrievals was attempted to find an answer to the problem. In the model's declarative memory, each exercise was not only connected to the correct answer, but also to incorrect answers. The more problem-answer connections there are, the harder it is to retrieve a correct answer.

Because of the associations with incorrect answers, retrievals could also result in an incorrect answer. Therefore, the associative strength of each successful retrieval was compared to a confidence criterion that was set at random in each trial. If the associative strength of the retrieved answer was lower than the confidence criterion, the answer was rejected and a new retrieval was started. Only when everything else failed, an alternative strategy would be applied. This alternative strategy was an algorithmic process, such as repeated addition, in which one multiplicand is added the number of times of the other multiplicand.

According to Siegler, the strategies that are initially used to solve the problem determine which problem-answer connections end up in the declarative memory. In turn, the problem-answer combinations in memory influence the strategies that are used to solve the problem. Thus, the errors children make early on in their development of the

Figure 1. The multiplication task in Math Garden. Responses are given using the keypad. On the bottom of the screen the current value of the problem is represented with coins.

multiplication skill are of great importance for the further development of the strategies they use.

In 1999, Lebiere implemented a combined memory and computational strategy model within the constraints of the cognitive architecture ACT-R. This model, similar to Siegler (1988), tries to retrieve the answer to an arithmetic problem, and, if that fails, uses repeated addition to calculate the answer. In the model by Lebiere, three sources of errors can be discerned. The first source of error stems from the use of repeated addition. For example, when solving the problem `3 x 8`, a student can accidentally do one additional step, resulting in `32`, or one step too few, resulting in `16`. Alternatively, an addition mistake in one of the steps can result in an answer such as `16 + 8 = 25`.

When retrieval becomes the main strategy to solve these problems, the errors made in repeated addition will still have an influence in the form of incorrectly stored combinations of a problem and its response. These lingering associations are the second source of error. The third source of error in the model is that of a partial match between stored information and to be retrieved information. For example, when the problem `3 x 5` needs to be retrieved, a mistake can be made by retrieving a similar problem, such as `3 x 4 = 12`, and giving the answer to that problem instead.

Current Study

Both Siegler and Lebiere did analyze the errors in their data. However, because of the relatively small datasets they used to fit their models, it was hard to examine the error patterns in a more systematical way:

> *"The patterns of errors for multiplication are quite rich, but harder to examine systematically because they take place over a wider range of values and display some characteristics (table errors, close misses, etc.), which are difficult to average over and plot together. For those reasons, let us concentrate on the pattern of errors for a single problem"* (Lebiere, 1999)

We will continue the work of Lebiere by looking in more detail at the errors in a large dataset that gathers the data of thousands of primary school students in all age categories. Although the proportion of errors is still low, the sheer amount of data makes the systematical analysis of these errors possible. Using these data, we can make finer grained assumptions about the strategies underlying these mistakes.

In this paper, we will start this endeavor by looking at three specific problems. First we will discuss the data and the different mistakes we find in the data, then we will propose strategies that can have led to these errors. We will implement these strategies in the cognitive architecture PRIMs (Taatgen, 2013) and discuss the similarities and differences between the results of the model and the data.

Task & Data

The data were gathered from "Math Garden" (see also: van der Ven, Straatemeier, Jansen, Klinkenberg, & van der Maas, 2015), an online computer application that is used by school children in the Netherlands to practice math and arithmetic. It offers problems that are adapted to the capabilities of the user, so that each problem has a reasonable chance of being solved (the default probability of correctly solving a problem is .75, this can be adjusted by the user). While the program contains a wide variety of different tasks, we will focus here on a standard multiplication task (see Figure 1).

Participants

Because we only have access to aggregated data from Math Garden, the specific distribution of participants is unknown. The users of Math Garden are Dutch primary school children with ages roughly between 5 and 13 years.

The Multiplication Task

The task we focus on in this paper is the multiplication task. In this task, a multiplication problem is presented on the screen (Figure 1). The student has to solve this problem within 20s. The answer is given by clicking on an on-screen keypad. Time is represented as a row of coins and every second a coin disappears from the screen. The coins that are left on the screen when the student has entered the answer are the score that is received, in the case of a correct answer, or lost, in the case of an incorrect answer. No points are awarded or lost when no answer is provided. This way of scoring is known as the 'High Speed High Stakes' principle (Maris & van der Maas, 2012). Students can decide for themselves how many trials they want to play and when they want to play.

Data

The data set we used was obtained on 25 May 2015 and contains the data of 8,489,703 attempts of 81 different problems (`1 x 1` – `9 x 9`). The overall percentage of errors in the full dataset was 10.42%. This is lower than the expected error percentage of 25% because late answers and answers in which the student asked for a hint were not taken into account. We will give a qualitative overview of the types of errors children make.

The most common errors often fit in one of the following categories:

1. The student has added the numbers instead of multiplying them. For example, `3 x 2 = 5`.
2. Operand related mistakes: the answer is consistent with the answer to a very similar problem. For example, `3 x 4 = 15`, which is the answer to the problem `3 x 5`, or `6 x 7 = 35`, which is the answer to the problem `5 x 7`.
3. Miss 1 errors: the answer is very close to the correct answer. For example, `3 x 5 = 14`, which is the

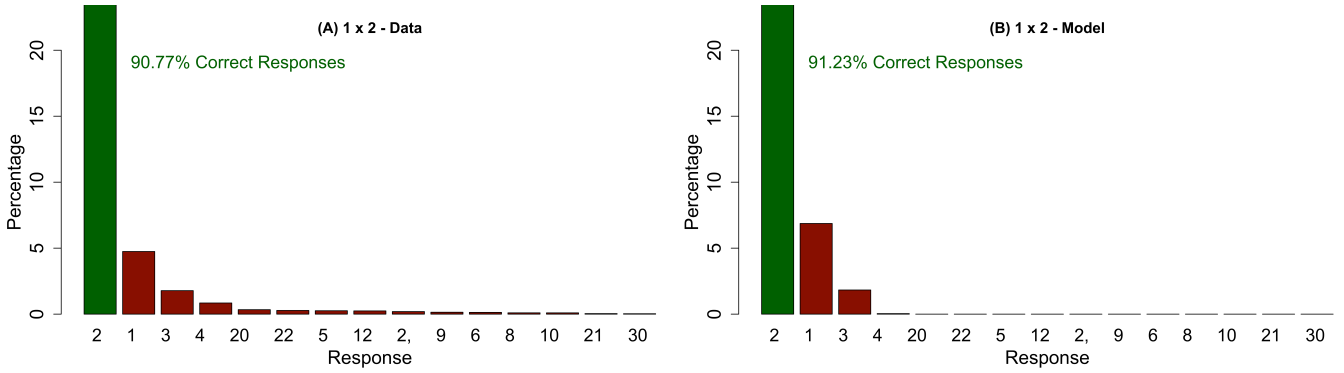


Figure 2. Responses given in the data (A) and by the model (B) after a simulation of 25 times 400 trials of the problem `1 x 2`.

correct answer minus one, `or 6 x 7 = 43`, which is the correct answer plus one.

Most of these mistakes can be explained by a mistake in the calculation or retrieval procedure. Interestingly, there are also mistakes that cannot easily be explained in the same manner. While previous models focused mainly on the first types of error, the goal of our model is to also explain some of the other mistakes. For ease of exposition we will focus on the errors in three different problems: `1 x 2`, `3 x 4`, and `9 x 6`. These problems are chosen because they fall in the first, second, and third tertile of the data, based on the Math Garden estimate of the average level of the students who have solved the problem correctly.

Explaining Multiplication Mistakes

We will discuss the five most commonly made mistakes for the abovementioned problems and how they are implemented in the model. Reading errors and input errors are outside the scope of the current model. We will start out by discussing the mistakes (see Figures 2A-4A) and hypotheses for the origin of these mistakes. Afterwards we will explain the setup of the model, and what the model can or cannot explain.

1 x 2

The problem `1 x 2` is an easy problem and was recorded 60,821 times in our dataset. In 91% of the cases the problem was solved, in the other 9% errors were made. The most common mistakes for the problem `1 x 2` are shown in Figure 2A.

The most common mistake is to give the answer 1, which is a pattern of behavior seen in all problems where one of the multiplicands is a `1`. There are several possible explanations for this mistake: (1) It could be caused by partial matching, instead of retrieving the required result for `1 x 2`, the result for `1 x 1` is retrieved. However, given that we also observe this pattern for `1 x 9`, this is unlikely. (2) The rule for multiplication by 0 can be overgeneralized. When a number is multiplied by 0, the answer is also `0`. Since the tables for 0 and 1 or often the first multiplication tables a student encounters, they might use the rule they have learned and used successfully for the table of 0 in the table of 1. (3) Finally, this mistake could be due to an incorrect application of the 1-rule. For all problems of the form `1 x N` it holds that the answer is `N`. If this rule is remembered or applied incorrectly, the result may be that the other multiplicand, the `1`, is considered to be the correct response.

The second, third, and fourth mistakes are `3`, `4`, and `20`. All of these answers fall into one of the categories mentioned earlier. `3` corresponds to either the addition of both multiplicands, an answer that corresponds to a similar problem, or miss 1 error. `4` corresponds to an operand related mistake, namely `2 x 2 = 4`. The most likely explanation for `20` is that the 1 is mistaken for a 10. This confusion is the third most made mistake in the table of 1, but it does not happen in any of the other tables. A likely explanation for the confusion is that the multiplication table for 10 is taught as one of the first multiplication tables and is therefore well known. The `1` is then easily confused with the `10`.

3 x 4

The problem `3 x 4` is a medium level problem; it was recorded 148,279 times in our dataset. 90% of the problems where solved correctly. In 10% of the cases an error was made. A specification of the errors can be found in Figure 3A.

The main mistake found in the data is `16`. `16` is the answer to `4 x 4`, an operand related error. It fits with the most common error categories we described before. It can either be caused by a retrieval that has gone wrong, either due to a previous mistake or due to a partial matching error. The other possibility is that the mistake is made because of a mistake in repeated addition; the student took one step too many in the calculation of the answer.

The next most common mistake is `9`. `9` is the answer to `3 x 3`, also an operand related error. Therefore, a similar explanation to the previous mistake applies.

The final three mistakes we will discuss here – `8`, `7`, and `15` – are made less often, but can be explained in a similar way. `8` is the answer to `2 x 4`, `7` is the answer to `3 + 4`, and `15` is the answer to `3 x 5`. In all these cases, this is

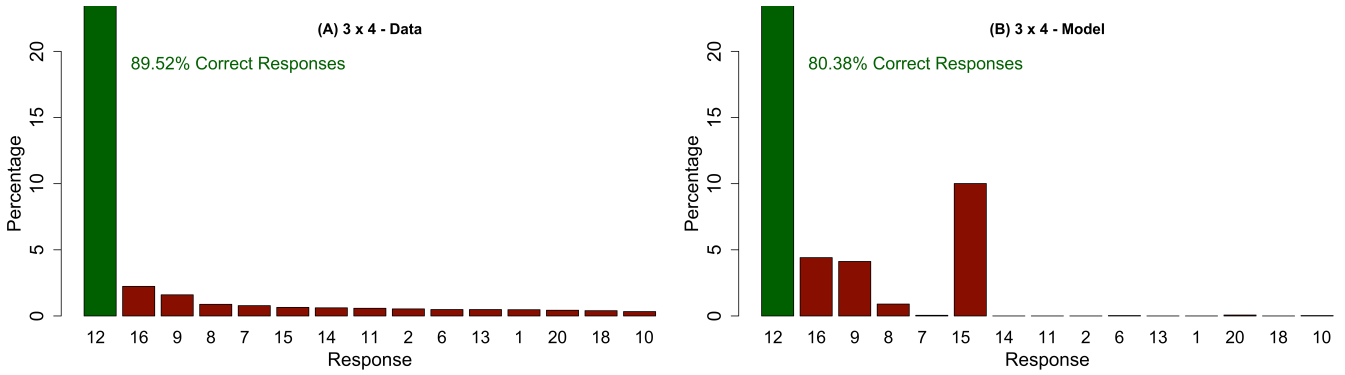


Figure 3. Responses given in the data (A) and by the model (B) after a simulation of 25 times 400 trials of the problem `3 x 4`.

either a mistake in the retrieval, a mistake in the repeated addition procedure, or a wrongly applied procedure (in the case of `3 + 4 = 7`).

6 x 9

`6 x 9` is one of the more difficult multiplication problems. It was recorded 90,145 times. 88% of the problems were solved correctly. The most common mistakes for `6 x 9`, as shown in Figure 4A, are `45`, `56`, `63`, `36`, and `53`. Of these responses, `45` and `63` are one step earlier and one step later in the table of `9`. In contrast, earlier and later steps in the table of `6` do not show up in the most common answers. This could indicate that students learn the commutative property relatively early, and apply repeated addition to the largest multiplicand. This takes fewer steps and therefore there is less opportunity for errors. Other evidence in the early learning of the commutative property comes from the mistakes made for problems and there exact opposite, such as `6 x 9` and `9 x 6`. In nearly all cases, the mistakes made for these two forms of a problem are exactly the same, and made with approximately the same frequency. Since the commutative property also holds for addition problems, it might be that the concept is learned early on for addition problems and transferred to multiplication problems.

`56` is the second most common mistake. This is most likely a mistake that is made because of errors in repeated addition or because the numbers `54` and `56` are very similar. `56` is also the correct answer to another difficult problem, `7 x 8`.

The fourth most common mistake, `36`, is either an input error for the response `63`, or the answer to `6 x 6`. Finally, `53` is probably an addition mistake. Independent of where in the sequence the mistake is made, it is very easy to arrive at a number close to `54`.

Overall, the mistakes made on the problem `6 x 9` seem to be less related to retrieval, and more representative of an algorithmic strategy, such as repeated addition.

A PRIMs Model of Single-Digit Multiplication

Our PRIMs model for multiplication was inspired by Lebiere (1999). The model has rules to retrieve the answer but also to compute the answer using repeated addition. The current model does not give an exhaustive fit of the data, but attempts to explain the most common cognitive mistakes. Errors in reading or input are outside of the scope of this paper. Model results are shown in Figure 2-4B.

PRIMS

PRIMs (Taatgen, 2013) is short for PRimitive Information processing eleMents. It is based on the ACT-R cognitive architecture (Anderson, 2007), but expands it by considering knowledge of tasks in a broader context. For our purposes this means that if PRIMs lacks task-specific knowledge, for example when it does not know a multiplication fact, it may try to determine the answer based on other skills it has, possibly, or even likely, producing an error.

PRIMs uses operators instead of production rules. The main difference is that PRIMs can use its operators for other tasks. This can be beneficial if other operators fill a knowledge gap, but it can also result in an error. Because of this, a model build in the PRIMs architecture does not only resemble someone who is already a perfect problem solver, but it can also account for the initial learning process.

Since PRIMs is based on the ACT-R cognitive architecture, it incorporates many of the mechanisms from this architecture. The declarative memory in PRIMs is based on the declarative memory in ACT-R: it contains chunks with related information. When a retrieval request is send to the declarative memory, the activation of each chunk is determined by the number of previous encounters with that chunk, and the time since those encounters.

The declarative memory of the current model also makes use of a mechanism called partial matching. In partial matching, chunks that do not completely match a retrieval query can also be retrieved. The activation of these chunks gets a mismatch penalty that is conversely proportional with the similarity to the requested chunk.

In PRIMs, the mismatch between numbers is calculated by taking the ratio of the smaller number to the larger number, minus one (based on Lebiere, 1999, p.48).

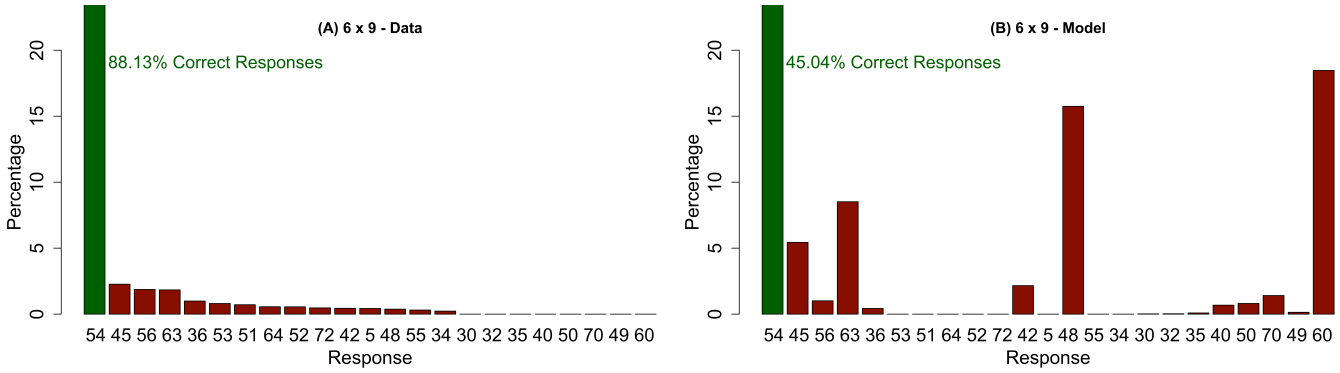


Figure 4. Responses given in the data (A) and by the model (B) after a simulation of 25 times 400 trials of the problem `6 x 9`.

The Model

For the current model, we have chosen to model the problems `1 x 2`, `3 x 4`, and `6 x 9`. For each problem, we assume that the model does not know the current problem at the start of the simulation, but it does know all addition facts and multiplication facts up to the current problem plus one. So for the problem `3 x 4`, all multiplication facts up till `4 x 5` are present in memory. This setup was chosen to simulate a large group of students that each know different problems.

The model in its current form uses standard parameter settings for ACT-R and PRIMs, to achieve these results, no parameter fitting was necessary.

The model first tries to retrieve the answer to the problem from memory; if retrieval fails the answer is calculated using repeated addition. However, a model that is purely based on retrieval and repeated addition cannot explain all the error patterns in the data. For example, the answer 1 is the predominant mistake made in the table of 1, but cannot be explained by a combination of retrieval and repeated addition.

1 x 2

The responses to the problem `1 x 2` are largely explained by the partial matching mechanism. The most common mistake however, `1`, is a mistake in the multiplication table 1, and it is likely that this mistake represents a wrongly applied strategy for solving problems of the form `1 x N` or `N x 1` – as explained above.

To explain this mistake, our model contains a specific strategy that can account for this mistake. This strategy represents the incorrect application of the rule for problems with the multiplicand `1`. When a problem has the multiplicand `1`, the other multiplicand is the answer to the problem. However, when this rule is incorrectly applied, it can easily result in the answer `1` being given, as we have seen in the data. This strategy is implemented in our model by a competition between production rules. When a problem in the form of `1 x N` is encountered, one of two rules can fire. Either the correct rule, which gives the correct response `N`, or the wrong rule which gives the incorrect response `1`. Over time the model will learn the correct rule through utility learning.

The results for the model on the `1 x 2` problem can be seen in Figure 2B. While this model captures the mistakes for the problem `1 x 2` relatively well, it still has some problems in fitting the error data for the larger problems, `3 x 4` and `6 x 9`.

3 x 4

The most common mistakes to the problem `3 x 4` can for the most part be explained by the partial matching mechanism. However, there is one error that stands out: `15`. We will explain the model's bias for this answer below.

Because the current model has only been applied to instance of specific problems, it does not take into account the frequency of exposure to previous problems. While it is known that problems with smaller multiplicands are practiced more often, this is not represented in our model. In other words, the current model does not incorporate the problem-size effect (e.g. Domahs, Delazer, & Nuerk, 2006).

Another effect that we do not reproduce is the tie effect. The tie effect is the relative ease of problems of the form `N x N`, such as `3 x 3`, which can explain the preference for the answers `16` (`4 x 4`), and `9` (`3 x 3`) over `15` (`3 x 5`), which is the main discrepancy between the model and the data. Lebiere (1999) did match this effect, by using spreading activation in the model. The information on the screen and the information in working memory spread activation to the chunks in declarative memory. In the case of a tie-problem, twice as much activation is spread to each slot in declarative memory, making it easier to retrieve these facts. The current model does not include spreading activation, and therefore does not account for the tie-effect.

Together, the absence of spreading activation and not taking into account the frequency of exposure can explain the strange peak we find in the model data for the problem `3 x 4`. While the first four mistakes are relatively well matched, the answer `15` is overrepresented in the model data. This is a side effect from the way the similarity between numbers has been implemented. A number is more similar to the number that follows it than to the number that

precedes it. For example, the similarity of the numbers 3 and 4 is −.25 (3/4 − 1 = −.25), while the similarity between the numbers 4 and 5 is −.2 (4/5 − 1 = −.2). This would be canceled out by the higher activation of chunks that are more practiced (smaller problems) and that are easier to retrieve (tie-problems). Both problems will be resolved in a future version of the model.

6 x 9

The pattern of errors for the problem 6 x 9 deviates significantly from the data. One reason for this is that the current model relies quite heavily on the retrieval strategy, while children probably also use an algorithmic strategy (van der Ven et al., 2015). The errors the model makes are therefore retrieval related errors, while the errors in the data seem to be caused by mistakes in repeated addition.

One possible explanation for this phenomenon is the number of times a student is exposed to a problem. As mentioned above, smaller problems, such as 1 x 2 and 3 x 4, are more prevalent in schoolbooks, but also outside of school. While the current model assumes the same frequency of exposure and a similar learning curve for both easy and hard problems, this may not necessarily be the case.

Discussion

In this paper we have presented a basic model for single digit multiplication. We have implemented and extended the model by Lebiere (1999) in the PRIMs cognitive architecture. Our goal is to explain the error patterns found in the Math Garden data. With our model we have shown that we can fit some of the most common errors found in the data. Furthermore, the model also seems to capture the learning process from making mistakes to achieving the correct responses, since it starts out giving incorrect answers, but gradually gives more correct responses. However, the current model has difficulties explaining the data of the harder problems, which seem to be solved using different strategies than easier problems.

Because the amount of data from Math Garden, we assume that the errors are not specific to this dataset. When we compare the data from Math Garden to, for example, the data from Siegler (1988), there seem to be discrepancies in the kinds of errors that are made and in the relative frequency of the errors. However, because the data in the study by Siegler only comprises a small subset of students of a small age group, we believe that the math garden dataset is better suited to use as a basis for modeling errors.

While the model is not yet complete in the sense that it captures all of the effects found in the literature, it can capture the error patterns that children make while doing easy single digit multiplication problems and is therefore a first step in the understanding of the strategies and misconceptions that lead to mistakes. Our goal is to build a model that can fully explain our current data with regard to errors and correct responses, and which is able to predict the outcomes on new problems accordingly.

The model we present here is still work in progress. Future endeavors will focus on incorporating the effects found in the data, such as the problem size effect and the tie effect. As suggested above, the model will benefit from the implementation of spreading activation. Furthermore, instead of a model of specific problems, the goal is to build a model that is exposed to the full range of multiplication problems will give a better indication of the relative importance of specific multiplication facts in memory.

Another goal is to show the relationship between the multiplication skill and other skills that are taught at school, such as arithmetic skills. The choice of the cognitive architecture is therefore not a coincidence: the PRIMs architecture is specifically developed to be able to systematically investigate interactions and relationships between different tasks. By investigating the relationship between different tasks, we hope to elucidate the existence of different strategies that are used to solve relatively simple tasks.

References

Anderson, J. R. (2007). How Can the Human Mind Occur in the Physical Universe?

Ashcraft, M. H., & Guillaume, M. M. (2009). Mathematical Cognition and the Problem Size Effect. In H. R. Brian (Ed.), *Psychology of Learning and Motivation, 51*. (pp. 121-151). Academic Press.

Domahs, F., Delazer, M., & Nuerk, H. C. (2006). What makes multiplication facts difficult. *Experimental Psychology*, *53*(4), 275–282.

Klinkenberg, S., Straatemeier, M., & Van Der Maas, H. L. J. (2011). Computer adaptive practice of Maths ability using a new item response model for on the fly ability and difficulty estimation. *Computers and Education*, *57*(2), 1813–1824.

Lebiere, C. (1999). The dynamics of cognition: An ACT-R model of cognitive arithmetic. *Kognitionswissenschaft*, *8*(1), 5–19.

Maris, G., & van der Maas, H. (2012). Speed-Accuracy Response Models: Scoring Rules based on Response Time and Accuracy. *Psychometrika*, *77*(4), 615–633.

Siegler, R. S. (1988). Strategy Choice Procedures and the Development of Multiplication Skill. *Journal of Experimental Psychology: General*, *117*(3), 258–275.

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological Review*, *120*(3), 439–471.

van der Ven, S. H. G., Straatemeier, M., Jansen, B. R. J., Klinkenberg, S., & van der Maas, H. L. J. (2015). Learning multiplication: An integrated analysis of the multiplication ability of primary school children and the difficulty of single digit and multidigit multiplication problems. *Learning and Individual Differences*, *43*, 48–62.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

The Sum of Two Models: How a Composite Model Explains Unexpected User Behavior in a Dual-Task Scenario

Marc Halbrügge (marc.halbruegge@tu-berlin.de)
Quality & Usability Lab, Telekom Innovation Laboratories
Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587 Berlin

Nele Russwinkel (nele.russwinkel@tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems
Technische Universität Berlin
Marchstraße 23, 10587 Berlin

Abstract

Maintaining cognitive control while pursuing several tasks at the same time is hard, especially when the current problem states of these tasks need to be represented in memory. We are investigating the mutual influence of a self-paced and a reactive task with regard to completion time and error rates. Against initial expectations, the interruptions from the reactive task did not lead to more errors in the self-paced task, but only prolonged the completion time. Our understanding of this result is guided by a combined version of two previously published cognitive models of the individual tasks. The combined model reproduces the empirical findings concerning error rates and task completion times, but not an unexpected change in the error pattern. These results feed back into our theoretical understanding of cognitive control during sequential action.

Keywords: Human Error; Memory for Goals; Working Memory Updating; Multi-Tasking; Threaded Cognition

Introduction

Multi-tasking and handling interruptions are very common in daily life. Both have been linked to reduced performance and increased error rates, even to road accidents (Altmann, Trafton, & Hambrick, 2014; Kujala & Salvucci, 2015). According to Borst, Taatgen, and van Rijn (2015), the central problem is the necessity to maintain several problem states at once. This can lead to interference between the respective memory traces, which manifests itself as error in one or several of the concurrently processed tasks.

Starting from this premise, we augmented an existing paradigm for error research during instruction following with a working memory updating (WMU) task. The WMU task should interfere with the primary task by a) periodically interrupting the user and b) additional memory strain. We hypothesized that this would result in increased error rates in the dual-task condition compared to a single-task baseline (Byrne & Bovair, 1997). In a previous study by Ament, Cox, Blandford, and Brumby (2010) using a comparable paradigm, high memory load was connected to higher error rates especially for device-specific tasks.

Another reason for the choice of the specific WMU task was the availability of a validated cognitive model of this task (Russwinkel, Urbas, & Thüring, 2011) that could be combined with the existing model of the primary instruction following task (Halbrügge, Quade, & Engelbrecht, 2016). This allowed to test the generalizability of the models to the new paradigm and at the same time provided the possibility to *quantify* the expectations from the memory interference effect that has *qualitatively* been layed out above.

This paper has two aims. First, we want to replicate the findings of Byrne and Bovair (1997), Ament et al. (2010) and others in an applied scenario. Second, we want to explore how much effort in terms of model development is needed to combine two existing cognitive models and how well the resulting model fits to the human data. Before presenting the empirical evidence, let us first clarify the basic concepts that are used in this paper.

Sequential Action and Procedural Error

Error research is usually concerned with failures on Rasmussen's rule-based level of action control (Rasmussen, 1983), i.e., well-learned routine activities like commuting to work or preparing breakfast. Errors on this level of control are relatively rare (below 5%), but pervasive (Reason, 1990). They are defined as the violation of the optimal path to the current goal, either by adding an unnecessary action (called *intrusion*), or by skipping an action (called *omission*).

A promising model for cognitive control during rule-based behavior is the Memory for Goals theory (MFG; Altmann & Trafton, 2002). It proposes that the steps that lead to the completion of a task are represented as subgoals in declarative memory (as defined within ACT-R, Anderson et al., 2004). Whether these subgoals can be retrieved and come into action depends on general memory effects like gradually decaying *activation*, *interference*, and *priming*. These effects are sufficient to explain important features of sequential action like postcompletion errors (Byrne & Bovair, 1997) and have been successfully implemented as computational cognitive models (e.g., Trafton, Altmann, & Ratwani, 2011; Tamborello & Trafton, 2015; Halbrügge, Quade, & Engelbrecht, 2015).

Postcompletion errors occur when an action sequence contains a final step *after* the goal already has been achieved, e.g., taking the original from a photocopier (the final step) after making a copy (the goal). What is so special about this final step? It does not contribute to the users' goal, but stems from the design of the device operated by them. This property of a task has been coined *device-orientation*, its opposite being *task-orientation* (Ament, Cox, Blandford, & Brumby, 2013;

Gray, 2000). Within the MFG theory, higher omission rates for device-oriented tasks can be explained by lower activation of the corresponding subgoals. While task-oriented subgoals receive priming from the overall goal, device-oriented subgoals do not. Previous modeling studies have shown that this differentiation is sufficient to explain disadvantages of device-orientation both in the completion time and the error domain (Halbrügge & Engelbrecht, 2014; Halbrügge et al., 2015).

Because of the downsides of device-oriented tasks, they are usually avoided during the design of user interfaces (UI). In case this is not possible, device-oriented tasks are often made *obligatory*, i.e., the users are forced to perform them by the application logic. Examples for this practice are the login button that users have to press after entering their credentials, or teller machines that return the card before delivering money. Making a step obligatory is quite effective. In a previous study, the expected error increase for device-oriented steps did only occur if the respective step was also non-obligatory (Halbrügge et al., 2015). Task necessity is therefore an important factor for the genesis of errors.

According to the MFG and Byrne and Bovair's (1997) preceding work, procedural error is caused by goal forgetting which in turn can be stimulated by high working memory load. In the context of this paper, we introduce memory load based on the WMU concept.

Working Memory Updating (WMU)

WMU is a task characteristic rather than a task itself. This concept describes the ability to maintain accurate representations of information changing over time (see Ecker, Lewandowsky, Oberauer, & Chee, 2010). Ecker et al. identified three putative phases of WMU – Retrieval (R), Transformation (T) and Substitution (S) of information.

The complexity of a WMU task can vary on two dimensions: *coordinative* complexity increases with the number of representations that have to be maintained at the same time while *sequential* complexity increases with update frequency (Mayr, Kliegl, & Krampe, 1996).

In the case of Ament et al. (2010), a low and a high memory demand condition was created by manipulating the coordinative complexity of the secondary (WMU) task. During the course of their experiment, the participants produced virtual doughnuts (main task) and had to count the amount of produced items of either one (low coordinative complexity) or two (high coordinative complexity) specific kinds of doughnuts.

For the present purpose, we considered the increase in complexity from one to several WMU targets as being too large. Instead, only one target (a pictogram) had to be counted and the sequential complexity was varied depending on how quickly the count had to be updated.

Experiment

We examined our assumptions using a kitchen assistant that has been created by computer scientists of TU Berlin as part of a smart home project (Feuerstack, 2009). The assistant aids in the preparation of meals by suggesting recipes, calculating ingredients and maintaining shopping lists. Its UI features all four possible combinations of device-orientation and task necessity, examples are given in Figure 1.

Method

Participants Twelve members of the Technische Universität Berlin paid participant pool took part in the experiment. There were four men and eight women, with their age ranging from 18 to 51 (M=33.7, SD=9.5). As the instructions were given in German, only fluent German speakers were allowed to take part. Written consent was obtained from all participants.

Materials The experiment was conducted in a neutral laboratory. A personal computer with 23" (58.4 cm) monitor with optical sensor 'touch' technology was used to display the interface of the kitchen assistant. Seven pictograms of common household interruptions (e.g., phone ringing, doorbell, baby crying; see Figure 2) served as stimuli of the WMU task. The stimuli were superimposed on the UI of the kitchen assistant using dedicated Javascript code running within the browser that displayed the assistant. All user actions were recorded by the computer system. The subjects' performance was additionally recorded on videotape for subsequent error identification.

Figure 2: Examples of the pictograms used in the experiment. Image credits: Baby © UN OCHA, CC-BY 3.0; Door, Cup with tea bag © Freepik, CC-BY 3.0

Design We applied a three-factor within-subjects design, the factors being device- vs. task-orientation, task necessity (non-obligatory vs. obligatory), and secondary task difficulty (none vs. onset to onset stimulus intervals 5s, 4s, 3s). User tasks were grouped into four blocks of eleven to twelve individual tasks. Each participant was randomly assigned to one of eight pre-selected block sequences so that block position and block succession were counterbalanced across participants as well. The secondary WMU task was always introduced after the completion of the first block and its sequential demand was gradually increased from 5s stimulus interval in the second to 3s in the fourth and last block. Each interval was split into equally long stimulus and and blank phases.

Procedure Every block started with comparatively easy recipe search tasks, e.g., "search for German main dishes and select lamb chops". Users would then have to change the search attributes, e.g., "change the dish from appetizer to dessert and select baked apples". The second half of each block was made of more complex tasks that were spread over

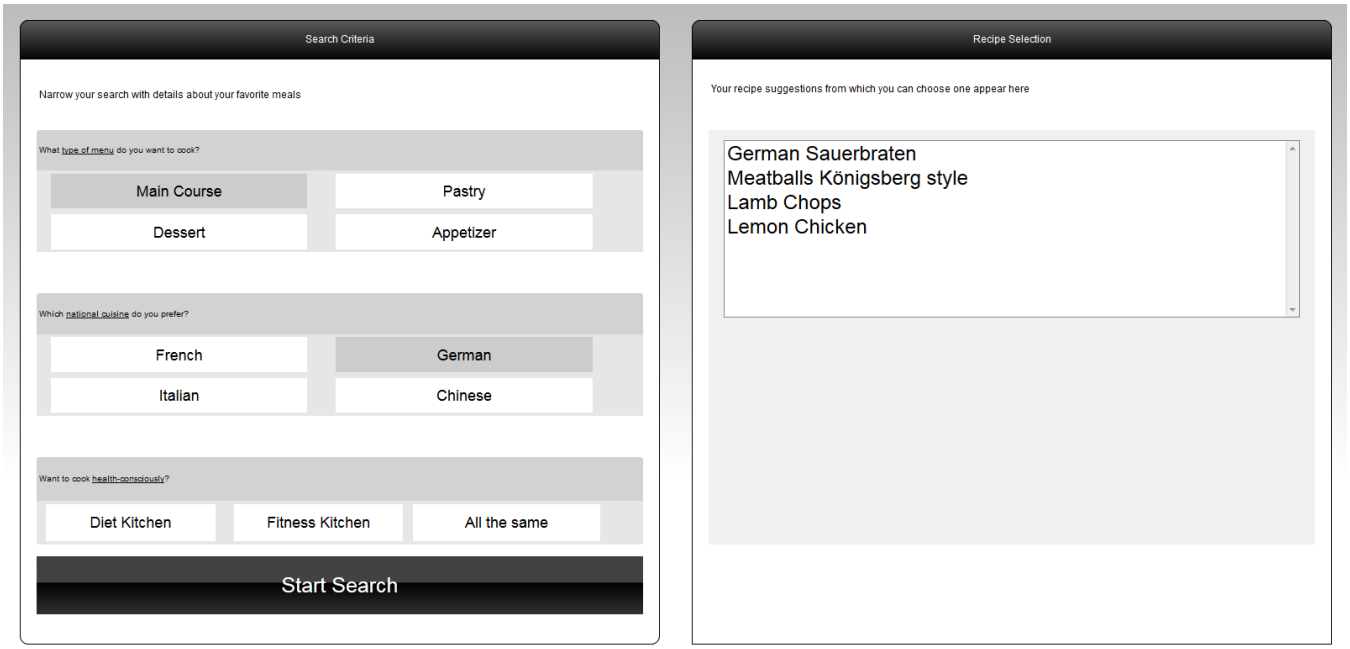


Figure 1: Screenshot of the English version kitchen assistant. Search attributes on the left are task-oriented and non-obligatory. The "Start Search" button is device-oriented and obligatory. The entries of the search results list on the right unhide subsequent options, they are therefore task-oriented and obligatory.

more screens of the interface and/or needed memorizing more items. The subjects either had to create ingredients lists for a given number of servings, or had to make shopping lists using a subset of the ingredients list, e.g., without salt and flour. All instructions were read to the subjects by the experimenter. Every individual trial was closed by a simple question the subjects had to answer to keep them focused on the kitchen setting, e.g., "how long does the preparation take?" During each instruction phase the complete screen was blanked (see Figure 3).

Figure 3: Sequence of screens within a single trial in the dual task condition.

In the dual-task condition, one of the seven WMU stimuli was selected as target for the current trial and presented to the participants after the instructions for the next trial had been given. Subsequently, the UI of the kitchen assistant was uncovered. WMU stimuli appeared in random order on the lower right of the screen and the participants had to count the number of appearances of the target stimulus. After the completion of the trial, the screen was blanked again and the participants were asked how often they had seen the WMU target. With an initial training phase and exit questions the whole procedure took approximately one hour.

Results

We recorded a total of 3464 user actions and 407 minutes of video. The system logs were synchronized with the videos and semi-automatically annotated using ELAN (Wittenburg, Brugman, Russel, Klassmann, & Sloetjes, 2006).

Recipe Task We recorded a 88 (2.5%) omissions and 133 (3.8%) intrusions. Contrarily to our assumptions, the error rate did not increase in the dual-task condition, nor when interruptions by the secondary task became more frequent. Adding the block (A–D in Figure 4) to a mixed logit model with task block and subject as random factors (Bates, Maechler, Bolker, & Walker, 2013) did not explain more variance ($\chi_3 = 1.30, p = .730$). Descriptively, the error rate even decreased while the memory updating task became harder.

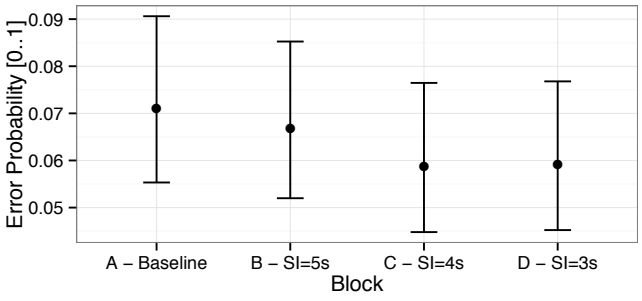


Figure 4: Error probabilities for the recipe task per experiment block. Error bars denote 95% confidence intervals based on the Agresti-Coull method.

There was a significant influence of the WMU task on the time needed to perform the recipe task. In the dual-task condition, participants needed approximately 100 ms longer per individual click (mixed model with click type[1] and subject as random factors, $t_{2695} = 2.31, p = .021$)

The influence of device-orientation and task necessity on errors was analyzed separately for omissions and intrusions (see Figure 5). Obligatory tasks led to fewer omissions than non-obligatory ones (logit mixed model with subject and task block as random factors, $z = -2.56, p = .011$). We observed fewer intrusions for obligatory tasks ($z = -4.16, p < .001$) and for device-oriented tasks as well ($z = -3.01, p = .003$). The significant interaction between both factors ($z = 2.41, p = .016$) is due to non-obligatory task-oriented actions (i.e., search attributes, left part of Figure 1) showing the highest intrusion rates.

Working Memory Updating Task Contrarily to our assumptions, the error rate in the memory updating task did not increase with the demand of task. Adding the block to a mixed logit model with task block and subject as random factors did not explain more variance ($\chi_2 = 0.12, p = .942$). Descriptively, we see a small increase, but the overall error rate is very high (see Figure 6).

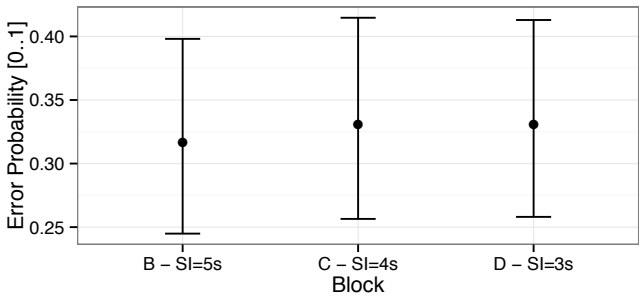


Figure 6: Error Probabilities per Block for the WMU Task. Error bars denote 95% confidence intervals based on the Agresti-Coull method.

Discussion

Interruptions are often used in error research as a means to increase the error base rates (e.g., Trafton et al., 2011; Li, Blandford, Cairns, & Young, 2008). In line with this thinking and based on the results of previous research (e.g., Byrne & Bovair, 1997; Ament et al., 2010), we expected that the increased memory load due to the WMU task would result in degraded performance in the main recipe task. But the empirical data tells a different story. While the participants needed more time to complete the recipe tasks, they did not make significantly more errors. Why is this the case?

First, the baseline error rate of 7% is already quite high, in particular compared to the 1.3% observed during a previous study using a similar paradigm (Halbrügge et al., 2015). This could be due to the blanking of the screen during the instruction phase that was added to the procedure in the present experiment. The blanking should have impaired the learning of the UI of the kitchen assistant. In previous studies, the participants could visually plan their actions during the instruction phase, while the current experiment demanded memorizing all subgoals without any visual reference.

Second, the high error rate of the WMU task suggests that the participants spent most attention on the recipe task. But as we only observed a very slight decrease of WMU performance with increased difficulty, this point remains unsatisfactory.[2] The analysis of the tasks based on the cognitive model presented below will provide additional insights.

The prolongation of the time needed to perform the recipe task in the dual-task condition is in line with the expected interference between both tasks. The real effect is probably underestimated by our analysis, because we found learning effects of approximately -50 ms per block in previous studies (Halbrügge & Engelbrecht, 2014). Assuming that learning still took place in the current study, it should have counteracted the prolongation effects of increasing WMU complexity.

Cognitive Model

Based on the well-established MFG theory, we proposed that having to perform two memory-intensive tasks would lead to more errors, but the data did not confirm our hypothesis. Does this disprove the theory, or was our understanding of it insufficient? In order to elaborate on the second option, we combined an existing MFG-based model of sequential behavior (Halbrügge et al., 2015) with an existing model of WMU (Russwinkel et al., 2011) using the threaded cognition extension of ACT-R (Anderson et al., 2004). The threaded cognition theory (Salvucci & Taatgen, 2008) assumes that task switching is not necessarily conscious behavior, but may emerge as concurrent tasks have to wait for cognitive resources (e.g., memory, vision) that are currently held by other tasks.

Recipe Task Model

The recipe task model extends on the MFG theory (Altmann & Trafton, 2002) by highlighting the importance of environmental cues during sequential behavior. Whenever the purely memory-based process as proposed by the MFG fails, the model reverts to a vision-based strategy that searches the environment for appropriate cues for the next action to take (see flowchart in Figure 7). This addition has been shown to be explain the effects of obligatory vs. non-obligatory steps with regards to omissions (Halbrügge et al., 2015), it has expanded MFG-based models to intrusion errors, and it has recently been confirmed by gaze data (Halbrügge et al., 2016).

For the current paper, the model was adapted to threaded cognition by adding extra checks for the current availability

[1] Four click type groups: same button, same group of buttons, different group of buttons, buttons on different UI screens; see Quade, Halbrügge, Engelbrecht, Albayrak, and Möller (2014) for reference.

[2] Unfortunately, Ament et al. (2010) give no results of the secondary task that could be used for comparison.

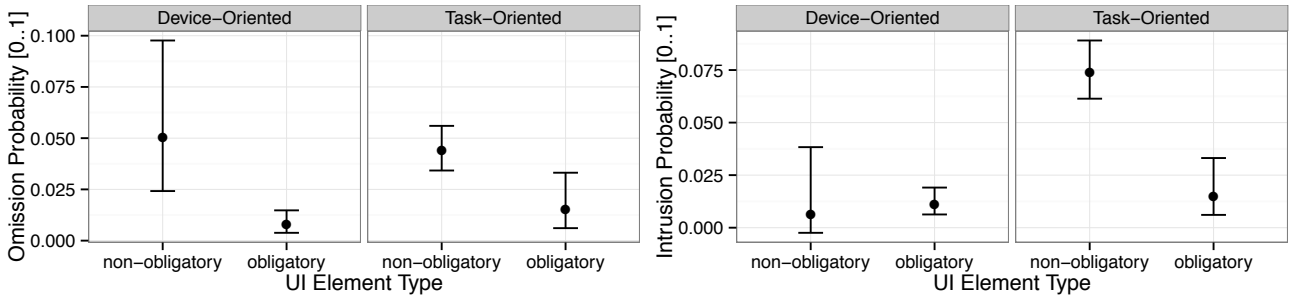


Figure 5: Omission and intrusion probabilities per UI element type. Error bars denote 95% confidence intervals based on the Agresti-Coull method.

of the declarative module to several productions. All numerical ACT-R parameters remained unchanged.

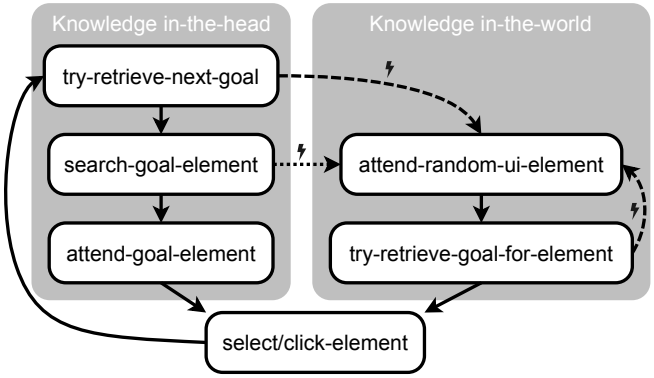


Figure 7: Simplified Flow Chart of the recipe task model. Dashed arrows denote retrieval errors, the dotted arrow denotes visual search failure.

Working Memory Updating Model

The code was adapted from an existing model that has been tested in different kind of tasks and settings (Russwinkel et al., 2011; Pape & Urbas, 2009; Russwinkel & Schinkmann, 2011). The WMU model uses a single representation (i.e., memory chunk) for each target that pairs it with its current count. This representation is manipulated using the three Retrieval, Transformation, and Substitution phases as proposed by Ecker et al. (2010). After its retrieval (R), the count slot of the chunk is updated (T). The resulting new representation is encoded in declarative memory (S), where it may interfere with older versions featuring outdated count values.

In the combined model, buffer stuffing is used to detect the visual targets of the WMU task. In case a new object is found at the right bottom of the screen and both the visual and the declarative modules are available, the model attends the new object and at the same time retrieves the most highly activated WMU count chunk. Because of activation noise, the declarative module may return an older copy of that chunk which subsequently leads to an error.

Goodness-of-Fit

The combined model[3] was run 500 times and all resulting errors and completion times were recorded. Contrarily to our expectations, but consistent with the empirical findings, the combined model does not show an increased error rate when the WMU task is present (odds ratio = 1.04, well within the empirical 95% CI from 0.62 to 1.20).

Regarding the effects of device-orientation and task necessity, the overall model fit is not good with R^2=.174 and RMSE=.029. This is mainly due to the unexpectly high intrusion rate for non-obligatory task-oriented steps (see Figure 8). When regarding only omissions, the fit is much better with R^2=.789 and RMSE=.014.

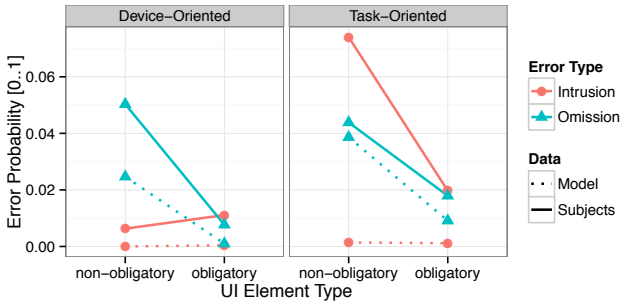


Figure 8: Model Predictions and Empirical Error Rates.

The combined model does show longer click times in the dual-task condition. Here, each click takes 120 ms longer on average, which is close to the empirical effect of 94 ms (95% CI from 14 ms to 174 ms).

Discussion

When the combined model performs both the recipe and the WMU task, the number of errors in the recipe task does not increase, it just takes longer to perform the individual actions. This means that our empirical results, although being unexpected, fit with the theoretical underpinnings presented above. Close inspection of the model traces shows that the

[3]The source code of the cognitive model is available for download at http://dx.doi.org/10.5281/zenodo.55224

model predictions are caused by both tasks demanding visual and memory ressources. Only during the motor phase of the recipe task model (i.e., when a click is performed, "select/click-element" in Figure 7), the WMU task can take over. This observation is also consistent with the high error rate of the WMU task during the experiment.

Combining both models using threaded cognition was less easy than expected. A central assumption of threaded cognition is that ACT-R buffers are shared between all running tasks. In our case, this made some states ambigous. Especially retrieval errors could not easily be attributed to the recipe or the WMU task (R phase of the WMU model vs. dashed arrows in Figure 7). This problem was solved by using the visual system. Retrieval errors are only attributed to the WMU task if the model visually attends a WMU target at the same time.

General Discussion and Conclusions

We have presented empirical data and a cognitive model of human performance and error in a dual-task scenario. Contrarily to our expectations, the dual-task condition did not lead to more errors, but longer task completion times, only. These results are nonetheless consistent with the Memory for Goals theory that had led to our expectations, as shown by the cognitive model simulations. The good fit is remarkable as no numerical parameter fitting was applied to achieve it. To our knowledge, this is also the first time that a MFG model has been combined with the threaded cognition theory.

The model has several limitations. First, the increased overall error rate, especially concerning intrusions, is not covered by the model. Second, the model does not show any specific visual behavior during the instruction phase, but only listens to the experimenter. Compared to previous studies that did not use a blank screen during the instruction phase (Halbrügge et al., 2015, 2016), the current data show relatively high error rates even in the single-task condition. This suggests that the human participants visually prepared their action sequence while listening to the experimenter in previous studies. More research is needed to elaborate on this point. We are therefore planning to examine the visual processing of the screen during sequence planning using eye-tracking.

As final remark we would like to highlight how the approach taken here exemplifies the benefits of computational cognitive modeling as a method. Because of the use of ACT-R as common denominator, is was possible to take two cognitive models created by different researchers and to combine them to something new that created new evidence and sparked new questions.

Acknowledgments

This work was partially supported by DFG grant MO 1038/18-1 ("Automatische Usability-Evaluierung modellbasierter Interaktionssysteme für Ambient Assisted Living").

References

Altmann, E. M., & Trafton, J. G. (2002). Memory for goals: An activation-based model. *Cognitive science*, *26*(1), 39–83. doi: 10.1207/s15516709cog2601_2

Altmann, E. M., Trafton, J. G., & Hambrick, D. Z. (2014). Momentary interruptions can derail the train of thought. *Journal of Experimental Psychology: General*, *143*(1), 215–226. doi: 10.1037/a0030986

Ament, M. G., Cox, A. L., Blandford, A., & Brumby, D. (2010). Working memory load affects device-specific but not task-specific error rates. In S. Ohlsson & R. Catrambone (Eds.), *Proceedings of the 32nd annual conference of the cognitive science society* (pp. 91–96). Portland, OR.

Ament, M. G., Cox, A. L., Blandford, A., & Brumby, D. P. (2013). Making a task difficult: Evidence that device-oriented steps are effortful and error-prone. *Journal of experimental psychology: applied*, *19*(3), 195. doi: 10.1037/a0034397

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological review*, *111*(4), 1036–1060. doi: 10.1037/0033-295X.111.4.1036

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2013). lme4: Linear mixed-effects models using eigen and s4 [Computer software manual]. (R package version 1.0-5)

Borst, J. P., Taatgen, N. A., & van Rijn, H. (2015). What makes interruptions disruptive? a process-model account of the effects of the problem state bottleneck on task interruption and resumption. In *Proc. chi 2015* (pp. 2971–2980). doi: 10.1145/2702123.2702156

Byrne, M. D., & Bovair, S. (1997). A working memory model of a common procedural error. *Cognitive science*, *21*(1), 31–61. doi: 10.1207/s15516709cog2101_2

Ecker, U. K., Lewandowsky, S., Oberauer, K., & Chee, A. E. (2010). The components of working memory updating: an experimental decomposition and individual differences. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *36*(1), 170.

Feuerstack, S. (2009). *A method for the user-centered and model-based development of interactive applications* (Doctoral dissertation, Technische Universität Berlin). doi: 10.14279/depositonce-2078

Gray, W. D. (2000). The nature and processing of errors in interactive behavior. *Cognitive Science*, *24*(2), 205-248. doi: 10.1016/S0364-0213(00)00022-7

Halbrügge, M., & Engelbrecht, K.-P. (2014). An activation-based model of execution delays of specific task steps. *Cognitive Processing*, *15*, S107-S110.

Halbrügge, M., Quade, M., & Engelbrecht, K.-P. (2015). A predictive model of human error based on user interface development models and a cognitive architecture. In N. A. Taatgen, M. K. van Vugt, J. P. Borst, & K. Mehlhorn (Eds.), *Proceedings of the 13th international conference on cognitive modeling* (p. 238-243). Groningen, the Netherlands: University of Groningen.

Halbrügge, M., Quade, M., & Engelbrecht, K.-P. (2016). Cognitive strategies in hci and their implications on user error. In *Proceedings of the 38th annual meeting of the cognitive science society.* (accepted)

Kujala, T., & Salvucci, D. D. (2015). Modeling visual sampling on in-car displays: The challenge of predicting safety-critical lapses of control. *International Journal of Human-Computer Studies*. doi: 10.1016/j.ijhcs.2015.02.009

Li, S. Y., Blandford, A., Cairns, P., & Young, R. M. (2008). The effect of interruptions on postcompletion and other procedural errors: an account based on the activation-based goal memory model. *Journal of Experimental Psychology: Applied*, *14*(4), 314. doi: 10.1037/a0014397

Mayr, U., Kliegl, R., & Krampe, R. T. (1996). Sequential and coordinative processing dynamics in figural transformations across the life span. *Cognition*, *59*(1), 61–90.

Pape, N., & Urbas, L. (2009). Testing a quantitative model of time estimation in a load-switch scenario. In A. Howes, D. Peebles, & R. P. Cooper (Eds.), *Proceedings of the 9th international conference of cognitive modeling.* Manchester, UK.

Quade, M., Halbrügge, M., Engelbrecht, K.-P., Albayrak, S., & Möller, S. (2014). Predicting task execution times by deriving enhanced cognitive models from user interface development models. In *Proceedings of the 2014 acm sigchi symposium on engineering interactive computing systems* (pp. 139–148). New York, NY, USA: ACM. doi: 10.1145/2607023.2607033

Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *Systems, Man and Cybernetics, IEEE Transactions on*, *13*, 257–266. doi: 10.1109/TSMC.1983.6313160

Reason, J. (1990). *Human error*. New York, NY: Cambridge University Press.

Russwinkel, N., & Schinkmann, M. (2011). Influence of working memory updating on time perception. In *9. berliner werkstatt mensch-maschine-systeme.*

Russwinkel, N., Urbas, L., & Thüring, M. (2011). Predicting temporal errors in complex task environments: A computational and experimental approach. *Cognitive Systems Research*, *12*(3), 336–354. doi: 10.1016/j.cogsys.2010.09.003

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: an integrated theory of concurrent multitasking. *Psychological review*, *115*(1), 101-130. doi: 10.1037/0033-295X.115.1.101

Tamborello, F. P., & Trafton, J. G. (2015). Action selection and human error in routine procedures. In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 59, pp. 667–671). doi: 10.1177/1541931215591145

Trafton, J. G., Altmann, E. M., & Ratwani, R. M. (2011). A memory for goals model of sequence errors. *Cognitive Systems Research*, *12*, 134–143. doi: 10.1016/j.cogsys.2010.07.010

Wittenburg, P., Brugman, H., Russel, A., Klassmann, A., & Sloetjes, H. (2006). ELAN: a professional framework for multimodality research. In *Proceedings of lrec* (Vol. 2006).

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Explaining inter-individual variability in strategy selection: A cue weight learning approach

Hrvoje Stojic[1] (hrvoje.stojic@upf.edu),
Henrik Olsson[2] (olsson@santafe.edu), **Pantelis P. Analytis**[3] (analytis@mpib-berlin.mpg.de)
[1]Department of Economics and Business, Universitat Pompeu Fabra, [2]Santa Fe Institute
[3]Center for Adaptive Behavior and Cognition (ABC), Max Planck Institute for Human Development

Abstract

Do people integrate all the information at hand when they make choices or do they employ heuristics that ignore some of it? Recent research indicates that people's behavior should and does depend on the statistical properties of the environments within which cognition operates. However, in a single environment there are always decision makers who rely on less effective strategies. The source of this inter-individual variation has not been identified yet. In this article we postulate that it can be largely explained by differences in the speed of learning. We designed an experiment where participants first made choices between three multi-cue alternatives and received feedback about their quality. In a second stage, they predicted the quality of alternatives without receiving feedback. The quality was a linear combination of cue weights and cue values. To employ heuristics the participants had to learn at least weight directions and ranks, while for the integrative strategy they needed to learn the cue weights. We find that participants who showed evidence of learning cue weights rather than the ordering performed well in the estimation task that followed decisions, with cue weight knowledge being strongly related to decision performance. Further, we find that differences in how fast participants learn the cue weights explain the variability in regards to what strategy they adopted within an environment.

Keywords: decision making; heuristics; cue weight learning; function learning; strategy selection.

Introduction

Consider the following problem: you want to decide which hotel to book for your next vacation and you have access to information such as the facilities of the hotel, average reviews, cleanliness etc. To make an educated choice you could weight and add all the information at hand for each alternative and then choose the one that achieved the highest score. This is a weighted additive strategy (WADD; Payne et al., 1993). Alternatively, you could compare the hotels according to the most important cue and choose the one with the largest cue value. If some alternatives are tied on the first cue, you could move to the next cue in the ranking until you reach a decisive cue and stop your search. This corresponds to a heuristic strategy called take-the-best (TTB; Gigerenzer & Goldstein, 1996). On average take-the-best would ignore most of the information, as your decision would often be based on a single cue. Researchers have investigated theoretically the conditions under which it is well-advised to rely on integrative strategies such as WADD or heuristic strategies like TTB (e.g., Hogarth & Karelaia, 2007, 2005; Martignon & Hoffrage, 2002). Empirically, however, there is a large inter-individual heterogeneity and substantial proportion of people still seem to use an inferior strategy (Bröder, 2003; Rieskamp & Otto, 2006; Pachur & Olsson, 2012).

Strategy performance primarily depends on the statistical properties of the relationship between cues and alternative quality. TTB fares well in comparison to WADD when the most informative cues are much more valuable than the less informative ones (Hogarth & Karelaia, 2007), or when the cue inter-correlations are high (Hogarth & Karelaia, 2005). In environments with binary cue values, when the weights of the cues with higher weight rankings are larger or equal to the sum of weights of the cues with lower rankings, TTB cannot be outperformed by WADD. When this property does not hold, a WADD model with well-calibrated weights is expected to outperform TTB. The former environments are called non-compensatory and the latter compensatory (Martignon & Hoffrage, 2002).

Several experiments have demonstrated that over time most people converge to the best performing strategy. For example, people tend to adopt TTB in non-compensatory environments and WADD in compensatory environments (Bröder, 2003; Rieskamp & Otto, 2006). Similarly, in non-linear environments, when none of the aforementioned two strategies performs well, many people employ memory-based exemplar strategies (Pachur & Olsson, 2012). Further, people prefer heuristic strategies over integrative strategies when they are under time pressure or when the cost of learning cue values is high (Rieskamp & Hoffrage, 2008).

Within a single environment, however, there is always a substantial portion of participants that use inferior strategies. For example, in a non-compensatory environment there are always participants that continue using WADD, or TTB in the compensatory environment. The source of this inter-individual variation has not been identified yet, although it is widely reported (e.g., Brehmer, 1994; Einhorn, 1970; Bröder, 2003; Rieskamp & Otto, 2006). Bröder (2012) provides a summary of existing research on inter-individual differences in adoption of TTB and WADD strategies. The only variable that shows some correlation is the intelligence score. TTB users in the non-compensatory environment tend to score higher on an intelligence test than WADD users, although the effect is rather small. None of the personality measures, such as the "Big Five", show a substantial correlation with strategy adoption. Similarly, motivational variables, cognitive styles, working memory capacity, and working memory load do not seem to influence adoption of TTB or WADD. Hence, the variation *within* an environment remains largely unexplained.

In this article we propose a solution to this puzzle. Strategies like TTB and WADD rely on cue weights. While in some

experiments participants are given the cue validity weights directly (e.g., Rieskamp & Otto, 2006), in most of them participants have to learn the weights (e.g., Bröder, 2003; Bergert & Nosofsky, 2007). Hence, besides figuring out which strategy to use, they also need to learn the statistical properties that are input to the strategies. Importantly, strategies differ with respect to the amount of knowledge they require about the validity weights. While WADD requires exact quantitative estimates, TTB only requires the ranking and directions. Under reasonable theoretical assumptions, heuristic strategies like TTB are largely insensitive to the gap between estimated and objective validity weights, while performance of WADD is heavily affected (Hogarth & Karelaia, 2007; Katsikopoulos et al., 2010). As a result, in many environments people can leverage WADD's improved performance only after some learning has occurred, and the estimated weights are relatively close to the objective ones. When coupled with usual individual differences in speed of learning, this explanation can address the observed variability in strategy selection. For example, in an environment favoring WADD, this leads to the prediction that slower learners will stick longer to the TTB heuristic, while faster learners will have more precise knowledge about the cue validity weights and will adopt WADD in greater numbers.

Our article suggests a novel approach in the study of decision making strategy by examining decision processes and cue weight learning in tandem. In our experiment, participants complete two tasks, a decision making and an estimation task. By adding an estimation task where participants make predictions about values of alternatives we can model their cue weight learning and infer the evolution of their knowledge about cue weights. Thus, we can identify the role of cue weight learning in strategy selection and test the predictions made above.

Method[1]

Participants

Seventy-eight participants (49 women, 29 men, $M_{age} = 21.8$, age range: 17–54 years), recruited from the Universitat Pompeu Fabra subject pool, took part in the study. They were paid a show-up fee of five euros and a performance dependent bonus of 6.8 euros on average. The experiment lasted 43 minutes on average.

Stimuli and procedure

The experiment consisted of two tasks: the participants first completed a decision making task and then an estimation task. In the decision task they repeatedly faced three alternatives, each described by the same four cues (Figure 1, left). The task was presented as a cheese game. Each alternative represented a cheese, the cues were "Lactic", "Acetic", "Casein" and "Texture", while the alternative values represented enjoyment units (EU).

[1]The raw data is publicly available on Figshare: http://dx.doi.org/10.6084/m9.figshare.1609680.

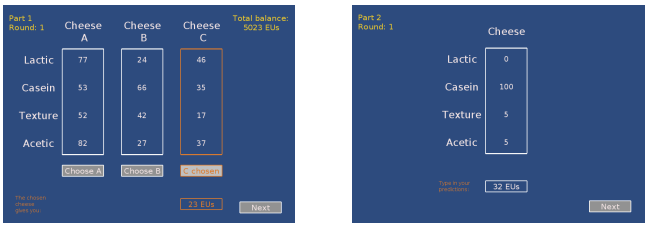


Figure 1: Left: Decision task. Right: Estimation task.

The criterion value, Y, of each alternative was a noisy linear combination of cue values and cue validity weights with weights fixed at 4,-3,2 and -1. These cue validity weights strongly favors WADD over TTB. Cue values were sampled from uniform distribution $U(10,90)$. A normally distributed error term, $e \sim N(0,30)$, was added to each alternative. We created 480 unique alternatives in this manner and allocated them randomly across 160 trials, three alternatives per trial. Cue inter-correlations were zero on average. The stimuli were drawn only once and all participants received the same stimuli. The earnings were determined by the criterion value Y of the chosen alternative, which was also shown as feedback in each trial.

After every 40 trials in the decision task participants answered questions that probed their knowledge about the cue weights. Following Speekenbrink & Shanks (2010), we asked them to rate the strength of the relation between each cue and the value of the cheese on a scale from -10 (highly negative) to 10 (highly positive). Questions for all four cues were shown on the same screen, in the same order that was used to present the stimuli.

In the estimation task participants received a single alternative in each trial and their task was to predict the criterion value (Figure 1, right). No feedback was provided. We incentivized truthful reporting by computing the payoff as a function of a difference between the prediction P and the criterion value, $200 - |P - Y|$.

The stimuli for the estimation task were generated with the same cue validity weights as in the decision task. We generated 20 alternatives for interpolation trials by drawing cue values from the same range as in the decision task, $U(10,90)$, and multiplying them with weights. We generated extrapolation trials in an analogous way by drawing cue values from two intervals at the extreme ends, $U(0,10)$ and $U(90,100)$ that have not been experienced during the decision task. After a single draw was made, trials were randomly ordered and all participants received the same set of stimuli.

In the decision task the participants were informed about the cues and the range of values they could take, and that they could use this information in making their choices. They were not told about the functional relationship between cue values and value of the cheese, nor that the weights differ for different cues. It was stressed that in each trial they would get three new cheeses that differ in their cue values. The estimation task was announced at the beginning in the instructions, but without specifying details.

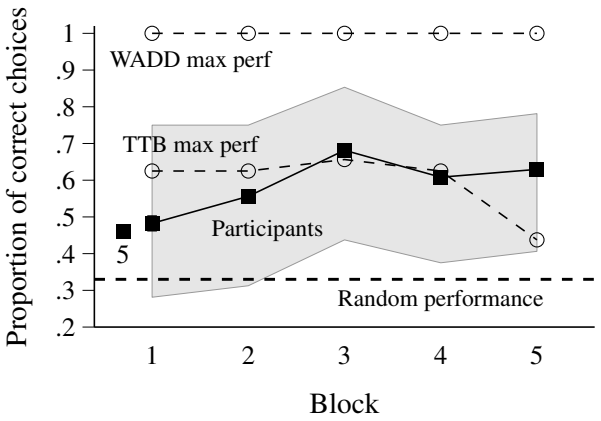


Figure 2: Mean accuracy over trial blocks. Each block result is a mean of individual means across 32 trials.

We told participants that it takes 60 minutes on average to complete the experiment. Each participant was presented with a unique random order of alternatives and cues. The four cue labels were also randomly attached to underlying cues separately for each participant.

Behavioral results

Choices in the decision task

Participants' performance, measured as percentage of correct choices per block, improved over time (Figure 2). Choice accuracy is much higher than the random level of 0.33 already in the first five trials (marked with number five in the figure), with 46% accuracy. People have a strong prior for positive linear relationships (Brehmer, 1994), which matches well the function that we used to construct the stimuli. Participants achieved a mean accuracy of 0.48 in the first block and by the end of the training phase they were close to choosing correctly the alternative with the highest criterion value two out of three times, 0.63. Although mean choice accuracy is similar to the accuracy achieved by TTB with ideal knowledge, 0.59 on average, the variance in individual choice accuracy curves is quite large. The shaded region around the mean performance indicates the range of accuracies, from 10^{th} to 90^{th} percentile. Hence, there are many individuals with accuracies far above what could be achieved with TTB.

Insight questions provide us with a first indication of how well participants have learned the cue validity weights. Previous research using such questions has shown that people have good insight into what they have learned (Speekenbrink & Shanks, 2010). Figure 3 shows mean ratings for all four cues. Participants got the relative ordering and directions right on average already after 40 trials and it got clearer as the training progressed. They learned that the second cue has a larger weight (although negative) than the third cue only at the end, and failed to detect that the fourth cue had a small negative weight. This is not surprising as negative linear relationships are more difficult to learn than positive linear ones (Brehmer, 1994). Although insight questions use an arbitrary scale and it is difficult to identify exact cue weights that participants have acquired, they do suggest that people learn more than ordering and directions. This is supported by changes in ratings over the course of the decision task, even though the ordering and directions were mostly established already after first time participants answered the insight questions.

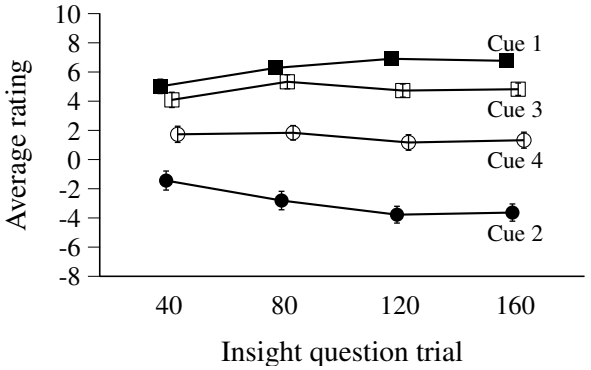


Figure 3: Average insight ratings across trials. Error bars represent standard errors of means across participants.

Predictions in the estimation task

We can also assess knowledge about cue validity weights by examining the performance in the estimation task. We computed mean absolute deviation (MAD) and correlation between participants' predictions and criterion values as a measure of performance. Mean MAD across participants is 120 (SD = 30.8), which means that on average predictions were 120 EU's away from criterion values. Mean (median) Spearman correlation is 0.63 (0.70; SD = 0.24). The participants are doing a good job in predicting criterion values of test items, but as expected, inter-individual variation in learning is substantial, with MAD ranging from 51 to 189. While most people are doing quite well, having very high correlations and low MAD's, some people do very poorly.

How would a decision maker that only learned the ranking of cues fare in the estimation task? Such a decision maker could take a mean of the criterion values experienced in the decision task and use it as a fixed prediction for all items in the estimation task. This is our baseline prediction performance. The MAD between baseline predictions and criterion values was 172, much larger than for observed MAD.

We get more complete insight by examining mean predictions across participants for each of the 40 items in the estimation task. Figure 4 shows that in the range of item values from about zero to 200, mean predictions correspond very closely to the criterion values. More deviations occur for more extreme values, with somewhat poorer predictions for extrapolation items than interpolation items. Importantly, predictions correspond much better to criterion values than baseline predictions. Thus, most participants do acquire more precise knowledge about cue validity weights, rather than only the ordering and directions.

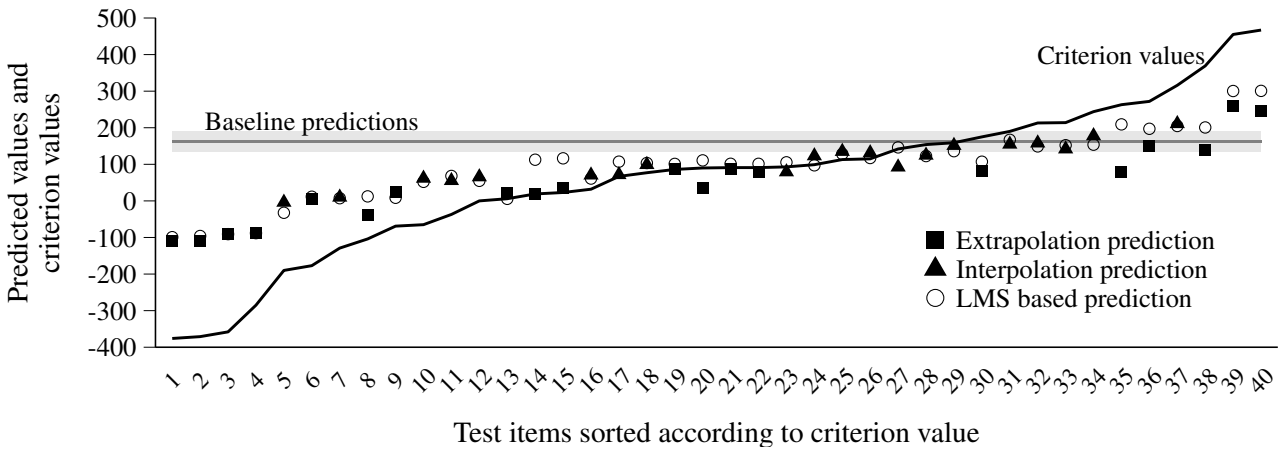


Figure 4: Mean prediction for each item in the estimation task. Black line that diagonally goes from lower left to upper right corner represents the criterion value of the items, while the gray horizontal line is the baseline prediction – mean value of the items experienced in the decision task.

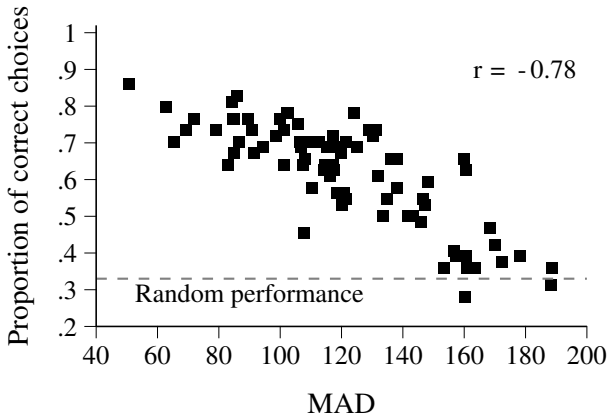


Figure 5: Relation between performance in the estimation task (mean absolute deviation (MAD) between predictions and criterion values) and the decision task (proportion of correct choices in the last two blocks).

We examine the relationship between individual performances in the two tasks to obtain model-free evidence that cue weight learning plays an important role in strategy selection. We find a strong relationship between choice accuracy in the decision task and MAD in the estimation task, as indicated by a Spearman correlation of -0.78 (Figure 5). This suggests that participants with good prediction performance know the cue weights well, which allowed them to employ WADD and achieve good decision performance. Surprisingly, many participants who had poor prediction performance also had decision performance far below 0.59 which is possible to achieve with very little knowledge for TTB in this environment. They either relied on WADD in spite of their poor knowledge or those participants simply paid less attention and performed close to random in both tasks.

Modeling

Next we turn to identifying the strategies used by each participant in the decision task. We first describe the cue weight learning model that will produce trial-by-trial predictions of participants' knowledge of cue weights. These weights will in turn be used in fitting TTB and WADD models to participants choice data. Finally, we will examine whether participants that were best fitted by TTB have less developed knowledge of cue weights than those best fitted by WADD, as predicted.

Modeling the cue weight learning

We used a least mean squares model to model the cue weight learning process (Gluck & Bower, 1988). The LMS model predicts the criterion value of an alternative on trial t as

$$P_t = \sum_{i=1}^{4} x_{i,t} u_{i,t},$$

where $u_{i,t}$ are cue utilization weights and $x_{i,t}$ are cue values of cue i in each trial t. Utilization weights are updated in every trial through the delta rule, based on a prediction error defined as the difference between the predicted criterion value, P_t and the true criterion value, Y_t, that a participant receives as a feedback in the decision task

$$u_{i,t+1} = u_{i,t} + \frac{\eta}{t^{\gamma}}(Y_t - P_t)x_{i,t},$$

where $0 \leq \eta \leq 1$ is a learning rate parameter shared by all four cues and $\gamma \geq 0$ is a decay parameter. We initialized the weights to $u_{i0} = 0$, $i = 1, ..., N$. Note that the cue weight learning process is based only on the alternative for which participants receive feedback, the rest is ignored by the LMS model.

We fitted two different versions of LMS model. LMS_d where both η and γ are free parameters and LMS where γ is set to 0. Parameters were initialized at the beginning of the decision task and in each trial cue values and criterion of the chosen alternative were used to update the weights. The weights from the last trial were used to make model based predictions in the estimation task. To estimate the model parameters we minimized the mean squared error between the participant's and model's predictions. The LMS model was fitted separately from the choice models.

Modeling the choices

Random Choice Model We used a random choice model (RCM) as a baseline. RCM predicts the same probability, .33, for each alternative.

WADD Model Our version of WADD linearly combines the cue utilization weights learned by the LMS model with cue values to produce predicted value of each alternative k in trial t

$$R_t^k = \sum_{i=1}^{4} x_{i,t}^k u_{i,t},$$

where $u_{i,t}$ are cue utilization weights learned by the LMS model based on trials $1:t-1$. WADD then deterministically decides by maximizing among the alternatives. To fit WADD to data we assume an additional "tremble" error. If a strategy produces a probability that alternative k is chosen, $P(C=k)$, then the probability of choosing k after taking into account the tremble error, ε, is given by

$$P(C_t = k;\varepsilon) = (1-\varepsilon) \times P(C_t = k) + \frac{\varepsilon}{3}$$

TTB Model Our version of TTB uses the cue weight information from the LMS model, $u_{i,t}$, to order the absolute value of the weights from the largest weight to the lowest, producing a ranking r_t. The ranking is done on absolute values because a strong negative weight is as predictive as a strong positive weight. TTB then chooses an alternative with the largest cue value of the most predictive cue according to ranking r_t. If values of the first cue according to the ranking are the same for all alternatives[2], TTB inspects the second cue and so on, until it finds a cue that discriminates between the alternatives. If no cue discriminates, a choice is made at random. If the deciding cue had a negative weight according to the u_t, cue values of all three alternatives were multiplied with -1, to maintain the correctness of the rule of choosing the alternative with larger cue value. Same as in the WADD model, we add a "tremble" error term to arrive at the final choice probability, $P(C_t = k;\varepsilon)$.

Modeling results

Table 1 shows the mean Bayesian Information Criterion (BIC) score across participants for LMS models and choice models. Both *LMS* and LMS_d fit the predictions equally well, both in terms of mean BIC (368 and 366) and number of participants best fitted (39 for both). However, LMS_d fits results better in a qualitative sense. It emulates better the insight questions results where most people acquire ordering and directions very fast. Hence, we used weights from LMS_d in the choice models. Moreover, LMS_d based predictions for estimation task items correspond closely to participants' predictions (Figure 4).

[2]Ties are rare in environments with continuous cue values, making this version of TTB quasi-equivalent to a single-variable strategy, which uses only the most important cue.

Table 1: Mean Bayesian Information Criterion (BIC) scores of models (standard deviation in the parenthesis), number of participants best fitted the model and mean parameter values.

Model	#	BIC	N	η	γ	ε
LMS	1	368 (25)	39	2e-5	-	-
LMS_d	2	366 (24)	39	2e-4	.61	-
WADD	1	283 (40)	56	-	-	.62
TTB	1	302 (40)	11	-	-	.74
RCM	0	355 (2)	11	-	-	-

Note. # = Number of parameters in the model; N = number of participants best fitted by the model; η = learning rate in LMS; γ = decay rate in LMS; ε = tremble error.

In terms of choice models, as expected, WADD has better mean BIC score (283) than TTB (302). Similarly, most participants were best fitted by WADD (56), followed by TTB (11) and RCM (11). As has been widely observed in previous studies, although it pays better to adopt WADD, and indeed most people do so, there is substantial inter-individual variability.There are substantial differences between the three groups. As expected, WADD users reached the highest accuracy, they were choosing the best alternative on average in 0.63 proportion of trials. TTB users performed worse, having a choice accuracy of 0.55. Although RCM users were the worst, reaching mean accuracy of 0.42, their performance is somewhat higher than the random level and they do exhibit some learning by the end of the training phase.

Next we examine our prediction that participants best fitted with TTB are those that learn slower and did not manage to arrive at sufficiently good utilization weights to switch to WADD. We plot the evolution of utilization weights estimated with the LMS_d model, separately for participants best fitted with each model (Figure 6). We see that WADD users have a well developed knowledge of all four cues, while TTB users have less developed knowledge. Notably, TTB users have very good estimates for the most important cue and do not distinguish that well between the other three cues. Their adoption of the TTB strategy is well justified by their subjective knowledge of the cue weights. RCM users' knowledge is very poor, capturing unmotivated or inattentive participants.

We can also examine estimated learning rate parameters of the LMS_d model. Learning rates are higher for WADD users than TTB users, and lowest for RCM users (Figure 6). Median learning rate for WADD users was 0.00015, while for TTB users it was lower for an order of magnitude, 0.000027. Median decay rates are correspondingly higher for the WADD users, 0.69, than for the TTB users, 0.54. Performance of TTB users in the estimation task ($M_{MAD} = 133$) was expectedly worse than that of WADD users ($M_{MAD} = 112$), but importantly, substantially better than of RCM users ($M_{MAD} = 155$) or baseline ($M_{MAD} = 172$). Similar differences can be seen in the insight questions results, with knowledge of TTB users evolving over time. This suggest that even a TTB user learns more than just the ordering and the direction of cues.

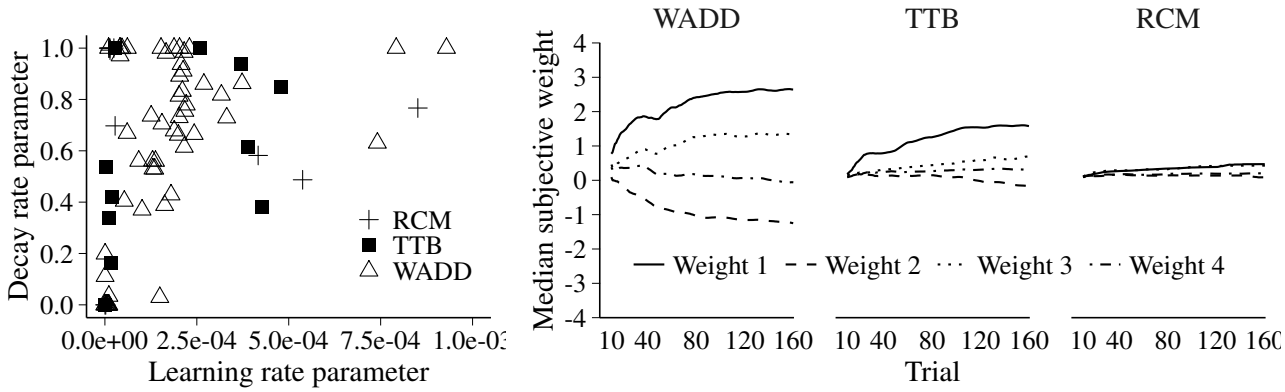


Figure 6: Left: Estimated learning η and decay rate γ parameters for the LMS_d model. Right: Smoothed median cue validity weights estimated with the LMS model for participants best fitted with WADD, TTB, and RCM.

Finally, we conducted a logistic regression with mean absolute difference between LMS obtained utilization weights in the last block and objective weight as a predictor of strategy use. We obtained a negative coefficient, as predicted, at a value of -1.466, with $p = 0.0139$ and $CI[-2.743, -0.367]$ (WADD users were coded as 1 and TTB users as 0, while RCM users were not included). In odds ratio terms, for one unit increase in mean difference, the odds of using WADD decrease by 76%. Odds of using WADD for the perfect knowledge (zero difference) is very high, 50.85, which amounts to a probability of 0.975. Although this outcome was already suggested by behavioral results illustrated in Figure 5, this analysis establishes the link between the knowledge of cue weights and strategy selection more clearly, in a model based manner. Since WADD users achieve greater decision performance, it explains the large correlation between estimation and decision performance seen in Figure 5.

Discussion & Conclusion

In our experiment participants differed in how fast they acquired knowledge of cue weights, and we predicted this heterogeneity to be responsible for the variability in strategy selection. Our results showed support for our predictions – WADD users had better developed knowledge of cue weights than TTB users and the performance in the estimation task is consistent with the strategy adoption. Our learning rate account suggests that, given time, TTB users would learn the weights sufficiently well and switch to the better performing WADD strategy.

Where do inter-individual differences in learning rates come from in the first place? These differences might be akin to traits like intelligence or personality factors investigated by Bröder (2012). This would require the learning rates to be stable across time and tasks within people. To our knowledge, there is no study that examines the stability of learning rates and is difficult to generalize beyond our task.

In our study we set out to test a specific hypothesis and to inform the debate on whether people are better described by the WADD or TTB model. We have to note that the models do not perform particularly well in our task. This can be witnessed in the high values of the ε parameter in Table 1, meaning that models on average predict the choices of the participants half of the time. Given our modest goals we did not try to look for models that would explain behavior even better. Our results, however, indicate that we should look for such models within the probabilistic rather than deterministic class of models (Bergert & Nosofsky, 2007).

Our results could be also explained if some participants first adopted TTB and as a consequence learned cue weights differently. With our current experimental design we cannot, unfortunately, determine the direction of the causal arrow. However, our evidence indicates that TTB users acquire more than ordinal information about cue weights and that this knowledge becomes more precise over time. This suggests that, if such interdependence exists, at most it slows down the learning. This evidence comes from three sources – the insight questions, the estimation task and the joint modeling of cue weight learning and decision making. The continuous evolution of our participants' knowledge of cue weights goes against the frugality and robustness justifications of TTB. The argument against using cue weights hinges on their vulnerability to overfitting – relying on ordinal information instead leads to better generalization. From our perspective, TTB and other heuristic strategies are used either due to cognitive limitations or when the structure of the environment is known better and these strategies are the rational thing to do (also see Davis-Stober et al., 2010; Davis-Stober, 2011).

In this decision-making task, our evidence suggests that learning the properties of the environment is predominant, and strategy selection is influenced by it. Different decision making tasks, however, may lead to distinct linkages between cue weight learning and decision making processes. Exploring the nature of these interactions opens an exciting direction for future research (see Stojic et al., 2015, 2016).

Acknowledgments

Hrvoje Stojic has been supported by FPU grant awarded by Ministry of Education, Culture and Sport of Spain (FPU12/05859) and by the Barcelona Graduate School of Economics.

References

Bergert, F. B., & Nosofsky, R. M. (2007). A response-time approach to comparing generalized rational and take-the-best models of decision making. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *33*, 107–129.

Brehmer, B. (1994). The psychology of linear judgement models. *Acta Psychologica*, *87*, 137–154.

Bröder, A. (2003). Decision making with the "adaptive toolbox": Influence of environmental structure, intelligence, and working memory load. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *29*, 611–625.

Bröder, A. (2012). The quest for take the best - Insights and outlooks from experimental research. In P. M. Todd, G. Gigerenzer, & the ABC Research Group (Eds.), *Ecological rationality: Intelligence in the world* (pp. 216–240). New York, NY, US: Oxford University Press.

Davis-Stober, C. P. (2011). A geometric analysis of when fixed weighting schemes will outperform ordinary least squares. *Psychometrika*, *76*(4), 650–669.

Davis-Stober, C. P., Dana, J., & Budescu, D. V. (2010). Why recognition is rational: Optimality results on single-variable decision rules. *Judgment and Decision Making*, *5*(4), 216.

Einhorn, H. J. (1970). The use of nonlinear, noncompensatory models in decision making. *Psychological Bulletin*, *73*, 221–230.

Gigerenzer, G., & Goldstein, D. G. (1996). Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, *103*, 650–669.

Gluck, M. A., & Bower, G. H. (1988). From conditioning to category learning: An adaptive network model. *Journal of Experimental Psychology: General*, *117*, 227–247.

Hogarth, R. M., & Karelaia, N. (2005). Ignoring information in binary choice with continuous variables: When is less "more"? *Journal of Mathematical Psychology*, *49*(2), 115–124.

Hogarth, R. M., & Karelaia, N. (2007). Heuristic and linear models of judgment: matching rules and environments. *Psychological Review*, *114*, 733–758.

Katsikopoulos, K. V., Schooler, L. J., & Hertwig, R. (2010). The robust beauty of ordinary information. *Psychological Review*, *117*, 1259–1266.

Martignon, L., & Hoffrage, U. (2002). Fast, frugal, and fit: Simple heuristics for paired comparison. *Theory and Decision*, *52*, 29–71.

Pachur, T., & Olsson, H. (2012). Type of learning task impacts performance and strategy selection in decision making. *Cognitive Psychology*, *65*, 207–240.

Payne, J. W., Bettman, J. R., & Johnson, E. J. (1993). *The adaptive decision maker.* New York, NY, US: Cambridge University Press.

Rieskamp, J., & Hoffrage, U. (2008). Inferences under time pressure: how opportunity costs affect strategy selection. *Acta Psychologica*, *127*, 258–76.

Rieskamp, J., & Otto, P. E. (2006). SSL: a theory of how people learn to select strategies. *Journal of Experimental Psychology: General*, *135*, 207–236.

Speekenbrink, M., & Shanks, D. R. (2010). Learning in a changing environment. *Journal of Experimental Psychology: General*, *139*, 266–298.

Stojic, H., Analytis, P. P., & Speekenbrink, M. (2015). Human behavior in contextual multi-armed bandit problems. In *Proceedings of the 37th annual conference of the cognitive science society.*

Stojic, H., Olsson, H., & Speekenbrink, M. (2016). Not everything looks like a nail: Learning to select appropriate decision strategies in multiple environments. *Submitted for publication*.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

EFFECT OF REWARD PREDICTION ERRORS ON THE EMOTIONAL STATE OF A MOBILE ROBOT

Vidullan Surendran (vus133@psu.edu)
The Pennsylvania State University, 234 Hammond Bldg.
University Park, PA 16802 USA

Lyle N. Long (lnl@psu.edu)
The Pennsylvania State University, 233 Hammond Bldg.
University Park, PA 16802 USA

Abstract

A goal-based dynamic action selection mechanism incorporating a model for emotions and temperament was developed for use with small and inexpensive mobile robots. A mobile robot was developed to test the action selection mechanism by recreating the scenario of an animal foraging for food while avoiding predators. Four emotions of anger, fear, happiness, and surprise were modelled which were affected by events such as finding food, encountering a predator, encountering a boundary wall, finding a safe area, and being in a state of low health. The model incorporated a reward prediction module that altered the effect of an event based on the error between when an event occurred and when it was predicted to occur. The model also included a decay term that resulted in the emotions returning to their steady state values unless there was continual reinforcement through the occurrence of events. The effect of differing temperaments on the emotions was studied by defining an irate temperament.

Keywords: Emotions; temperament; affective computing; robotics; action selection.

Introduction

Affective computing seeks to understand and develop systems that can recognize and simulate human emotions. R.W. Picard (2000) highlighted the importance of the field by exploring neurological studies that indicated human cognition was intrinsically linked with emotions. She also argues that that the development of affective computing is critical to advancing emotion and cognition theory.

Breazeal and Brooks (2005) considered cognition and emotions to be two distinct systems that evolved in intelligent creatures under social and environmental processes to aid in optimal functioning. Cognition is deemed to be responsible for interpreting the world whereas emotions are deemed to be responsible for evaluating the value of events. Emotions thus help prioritize concerns while minimizing distractions.

The simulation of human emotions is greatly complicated by the fact that there is no accepted model that explains and predicts the wide range of emotions we experience. There still is no consensus in the literature on the number of base emotions. Paul Ekman (1999) proposes a list of 15 emotions, each representing a family of emotions. A study on dynamic facial expressions of emotion by Jack et al. (2015) challenges this notion by suggesting that basic emotion communication comprises fewer categories. It is clear that our understanding of the subject is still in its infancy.

A study on momentary subjective well-being by Rutledge et al. (2014) resulted in a model of happiness referred to as the 'Happiness Equation' in popular culture. They showed that momentary happiness in response to a probabilistic reward is explained by the combined influence of the reward expectations and the prediction errors from the expectations; not by current task earnings as one would naively conclude. Long et al. (2015) and Long (2015) adapted this model to simulate the eight 'universal' emotions of fear, anger, sadness, happiness, disgust, surprise, trust, and interest in cognitive mobile robots. The emotion and temperament engine they developed was incorporated into SS-RICS which is a cognitive architecture developed at the Army Research Laboratory (Troy D. Kelley, 2006). Surendran (2015) built upon this emotional model by incorporating a reward prediction error component with a focus on developing a model that could be executed with limited computational resources. This model did not use SS-RICS and implemented a new action selection mechanism (ASM) capable of running on a small mobile robot with a Raspberry Pi processor.

Test Setup

A client-server architecture was used with an autonomous agent acting as the client, and a computer running the ASM and affective model acting as the server. A robotic platform was developed specifically for this study with an emphasis on a form factor under 400 cm^2 and reduced cost. It was based on the Raspberry Pi microprocessor and is capable of image processing at an average of 5 frames a second, orientation sensing, collision detection through infrared sensors, and battery operation for an hour.

The behavior and emotional state changes of a small rodent foraging for food while avoiding predators was chosen to be simulated. Differently colored balls having a diameter of 10 centimeters were used to represent food sources (green), predators (red), and safe areas (purple). The agent was placed inside an enclosed space containing the three types of balls which were randomly placed. One of the purple objects representing a safe area was chosen as the starting position of the robot.

The behavior that was simulated can be described as follows. The mobile robot 'rests' at a safe area until its health

decays below a threshold level triggering a search for food; this simulates hunger. The robot then searches the surrounding area until it identifies a green ball which it tracks towards. It then 'consumes' the food by being in close proximity (within 5 centimeters) of the ball. The satiation of hunger is simulated as a time dependent increase in the robot's health for as long as it stays in close proximity to the object. When its health has been completely recharged it then seeks out a safe area to 'rest' at, until its health drops below the threshold triggering a repeat of the cycle. Throughout the simulation, the mobile robot constantly avoids red danger balls and collisions with the boundary walls. Whenever a danger is identified, the robot abandons its current task and instead takes evasive manoeuvers to avoid the danger. Only stationary predators were considered in this study. A linear decrease in health with respect to time was assumed in this study. It is to be noted that the framework allows one to implement more complex models of health and consumption.

Action Selection Mechanism (ASM)

In order for the system to be autonomous it had to decide what to do. In order for it to be successful it had to intelligently select appropriate actions based on external and internal stimuli. Dynamic planning methods compute the next action to be taken based on the current internal and external state. This type of ASM is ideal when limited computational resources are available. On the other hand, to replicate the behavior required a goal driven architecture was used (Brom & Bryson, 2015). To this end a new hybrid architecture was implemented in the final system.

The behavior implemented was decomposed into goals and subgoals. Goals were namely finding food, finding a safe area, avoiding danger and resting. These goals were then decomposed into subgoals such as finding a ball, tracking a ball, eating a ball, avoiding a ball, etc. Each of these subgoals was associated with a dynamic plan that the ASM selects in order to accomplish the goal. This modular structure permitted reuse of plans as the goals had similar subgoals. Condition-action rules similar to those used in expert systems were used to implement the dynamic plans. Rules defined the 'knowledge' the system possessed and in this system are either factual or procedural. Factual knowledge as the name suggests consists of facts such as the color of a ball defining if it relates to food, danger, or a safe area. Procedural rules govern how actions the robot take will be carried out. Conflict resolution in case of competing goals is handled using priorities with certain goals having a higher priority. For example, if a danger ball is seen in front of an objective ball, the robot would avoid the danger instead of moving towards the objective.

An event handling function and an interrupt mechanism implemented in software handle flow control. The event handler executes the appropriate plan based on the current goal, subgoal, and system state. The interrupt handler allows the currently executing goal to be paused when a goal with a higher priority is to be executed. Once complete, it reloads the previous goal and subgoal.

Time Step Each time step is defined to be one cycle of the mobile robot sending data to the command center and receiving control commands. This was chosen over time as hardware limitations prevented real time image processing and consequently governed the rate of data transmission. This ensured consistency as otherwise a slower system would experience quicker decays in health and emotions due to processing time. If enough processing power is available, the framework allows clock time to be used instead.

Emotions Engine

The computational model obtained by Rutledge et al. (2014) is shown in Equation (1). In the model CR represents certain rewards, EV their expected value and RPE the reward prediction error.

$$Happiness(t) = w_0 + w_1 \sum_{j=1}^{t} \gamma^{t-j} CR_j + w_2 \sum_{j=1}^{t} \gamma^{t-j} EV_j + w_3 \sum_{j=1}^{t} \gamma^{t-j} RPE_j \quad (1)$$

Long et al. (2015) modified this model as shown in Equation (2) to incorporate emotions and temperaments into cognitive mobile robots. The winner-take-all approach they implemented meant that the emotion with the highest value was considered as the robot's emotional state. In this model R_{ij}^{+} represents positive reinforcements while R_{ij}^{-} represents negative reinforcements.

$$Emotions(t)_i = w_{0_i} + \sum_{j=1}^{t} {\gamma_i}^{t-j} (w_{1_i} R_{ij}^{+} + w_{2_i} R_{ij}^{-}) \quad (2)$$

The emotional model used in this research study is shown in Equation (3) and incorporates the RPE term from equation (1) into equation (2).

$$Emotion(t)_i = w_{0_i} + \sum_{j=1}^{n} \gamma_i^{(t_f - t_j)} w_{1_i} R_j + \sum_{j=1}^{n} \gamma_i^{(t_f - t_j)} w_{2_i} RPE_j \quad (3)$$

The positive or negative effect of an event on the emotion is represented by the term R_j. The reward prediction error is represented by the term RPE. w_0, w_1, and w_2 are weighting factors. γ is a decay factor that governs the impact of past events on the current emotional state. Together these values define the temperament of an agent.

Emotions and Temperament Constants

A distinction has to be made between temperaments and emotional states. Temperaments are considered to be biologically based and derived from genetic predispositions, maturation, and experience. They are expected to be relatively stable over time. Emotions from a functionalist approach are defined as a person's readiness to establish, maintain or change one's relationship to his or her changing circumstances. In contrast to temperamental variability, emotional reactions can be enduring or brief (Thompson & Winer, 2015).

Four emotional states of anger, fear, happiness, and surprise deemed to most likely be affected by the test scenario were modelled in this study. Each emotion is assigned weighting factors and a decay factor that correspond to Equation (3) as shown in Table 1. To better illustrate the system, the weighting factors were experimentally selected to result in significant changes in emotional values within a range of ±50. The steady state value (w_0) of the emotions was set as zero for the same reason. Future work will address how to optimize these constants to make the robot as effective as possible.

Table 1: Emotion constants

Emotion	w_0	w_1	w_2	γ
Anger	0	1.9	0.15	0.92
Fear	0	1.9	0.15	0.92
Happiness	0	2.3	0.15	0.92
Surprise	0	1.7	0.15	0.88

Table 2: Emotion modifiers

	Anger	Fear	Happiness	Surprise
Danger encountered	0	+12	-5	+10
Found Food	-1	-1	+5	0
Returned to safe area	-5	-5	+5	-5
Wall encountered	+3	0	0	+5
Health too low	0	+2	-2	0

Components

The emotions engine is made up of three major components; short term memory, the RPE module, and the command modifier.

Short Term Memory The emotional state of an agent is deemed to depend on its internal state and external stimuli such as events. Events are defined as interactions with the environment that result in a change of the ASM goal or subgoal. Table 2 lists all defined events along with their reward values. The emotions engine records registered events, the reward value for the event, and the time steps taken to complete the ASM goal that led to it. Due to the decay component (γ) of the emotional model, it was found that a memory length of six events was optimal as the exponential decay meant that the value of any previous events was negligibly small.

Reward Prediction Error (RPE) Module Momentary subjective well-being was found to depend not only on an event but errors in predicting the occurrence of said event. Unexpected events have a higher impact on the value of an emotion (Rutledge et al., 2014). Due to a lack of long term memory and learning capabilities, the average number of time steps taken for an event to occur in the past is used to predict future expectations. The difference between this prediction and the actual time taken for the event to occur is considered to be the RPE. At the start of an agent's life each type of events is predicted to take 25 time steps to occur. An exception is the 'Health too low' event which is considered to take 300 time steps to occur. These values were chosen as they were found to be the average number of time steps taken in the majority of tests. As events of a type occur during operation, the prediction for that event is updated.

Command Modifier Data stored by the short term memory module is sent to the RPE module to calculate the RPE. Using Equation (3) the instantaneous value of the four emotions are then calculated and in combination define the emotional state of the robot. These calculations are carried out every time step regardless of an event being registered. The command modifier allows the emotions engine to modify the ASM commands generated and the internal state of the agent, based on the current emotional state. Conditional logic and statements are used for this purpose. In this study we have chosen to apply the following modifiers,

- $Speed = current\ speed + value_a + value_f - value_h$

Where $value_a$, $value_f$, and $value_h$ denote the numeric values of the emotions anger, fear, and happiness respectively. This modifier increases the speed of the robot when the value of anger or fear increases and decreases it when happiness increases. This shows how all emotions can compete against each other to affect behavior.

- $if\ speed < 90\ then\ speed = 90$

This modifier sets a lower bound on the speed of the robot.

- $if\ value_f > 40\ then\ randomly\ stop\ the\ robot's\ motion$

This simulates a timid robot by randomly sending it stop commands if the value of fear becomes greater than 40. The is similar to the tendency of an animal to momentarily freeze when frightened.

- $if\ surprise > 40\ then\ make\ a\ 530°\ turn$

This simulates a surprised robot by making it turn around if the value of surprise gets larger than 40. This is similar to an animal being startled and a robot spinning around was not only amusing, but easily observable.

Results

The effects of the different type of events, the RPE module, and varying temperaments were considered. The test scenario pictured in Figure 1 consisted of two safe balls, three food balls, and three danger balls. It was run for 300 time steps.

Figure 1: Test Scenario

Events are signified on the plots using the letters shown in Table 3. A negative RPE denotes an unexpected event while a positive RPE denotes an event that was predicted to occur earlier than observed. Table 4 lists all events that were registered along with their time steps and the calculated RPE.

Table 3: Events and their signifiers

Event	Denoted by
Danger encountered	a
Found Food	f
Returned to safe area	b
Wall encountered	w
Health too low	h

Table 4 Time steps, Event type, and RPE value of all events

Step	**22**	**23**	**24**	**35**	**43**	**51**	**59**
Event	f	f	f	w	w	w	w
RPE	-13	-5	-1	10	-22	-11	-5
Step	**85**	**92**	**107**	**146**	**158**	**181**	**204**
Event	a	w	w	a	w	w	w
RPE	60	23	-6	6	33	-11	-4
Step	**214**	**217**	**225**	**231**	**241**	**242**	**248**
Event	b	w	w	w	f	w	f
RPE	164	-14	-11	-7	12	2	13
Step	**249**	**255**	**256**	**260**	**266**	**294**	
Event	w	f	w	f	f	w	
RPE	-3	14	-1	12	12	31	

Effect of Events on the Emotional State

Figure 1 shows the variation in the individual emotions whereas Figure 2 shows the variation in the robot's speed and its health. From the data we can infer that, once the health dropped below the threshold of 75 at time step 10, the agent began searching for food and found it at time step 22, that is in 12 steps. The RPE led to a larger spike in happiness and a large reduction in robot speed. During its search for a safe area it had to avoid the boundary walls, food balls, and the danger balls each of which affected the emotions as seen in the previous scenarios. The robot finally found a safe area at time step 214. The initial prediction by the RPE module was that a safe area would be reached in 25 time steps. In this case since that led to an RPE of 164, instead of an increase in happiness and reduction in anger, surprise, and fear, we see the opposite effect. Since the robot took a large number of time steps to return to a safe area, when it did its health had dropped below the threshold of 75. This caused it to immediately start searching for a food ball. At approximately time step 241 there was an anomaly where the robot incorrectly identified a food ball as a boundary wall momentarily before correctly recognizing it as a food ball.

Effect of the RPE Module

Figure 3 shows the effect of the RPE on the emotional state value of anger. Using Table 4 we study some key events that illustrate the effect of the RPE module:

Food event at time step 22 (RPE of -13) This event happened sooner than predicted by 13 time steps. The effect of the RPE module was a greater reduction in anger than would have been without it.

Wall event at time step 35 (RPE of +10) This event happened later than predicted by 10 time steps and thus its impact was reduced. This is seen as anger values not being as large as they would have without the RPE module.

Wall event at time step 43, 51, 59 (RPE of -22, -11, -5) We observe that the impact of the wall at step 43 is greater than that at step 51 and so on. When the RPE module was disabled, the impact of all 3 wall events was exactly the same.

Safe area ball at time step 214 (RPE of 164) The extremely large prediction error meant that instead of a decrease in anger that would have been expected when the robot returned to a safe area, there was an increase in anger. Basically the opposite of the effect the event would have had on the emotions had the RPE module been disabled.

We can thus conclude that the RPE module is crucial in modelling the psychological effect of expectations governing the effect of an event. Without it, an event would affect the emotional state exactly the same way regardless of when it occurs.

Effect of Varying Temperaments

Temperaments are defined by the weighting factors shown in Table 1.

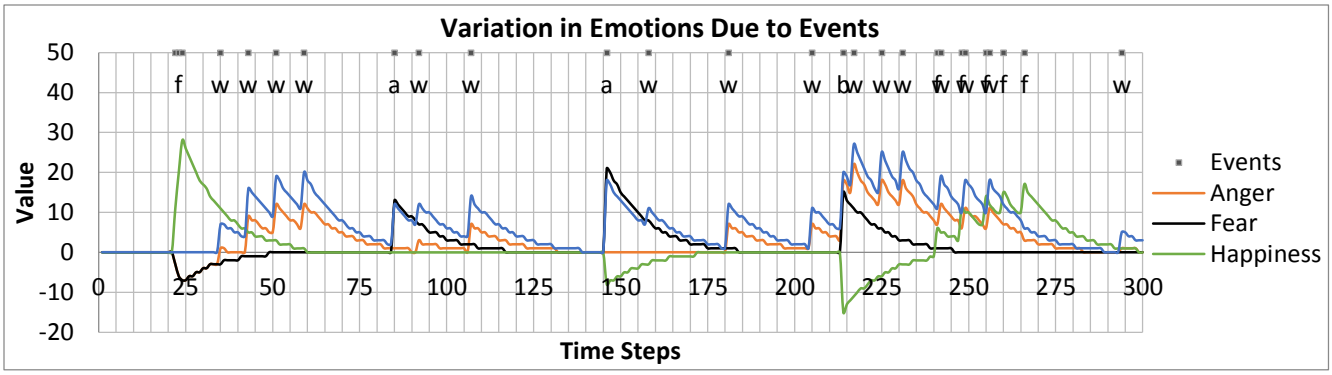


Figure 1: Shows the variation in the values of the four emotions

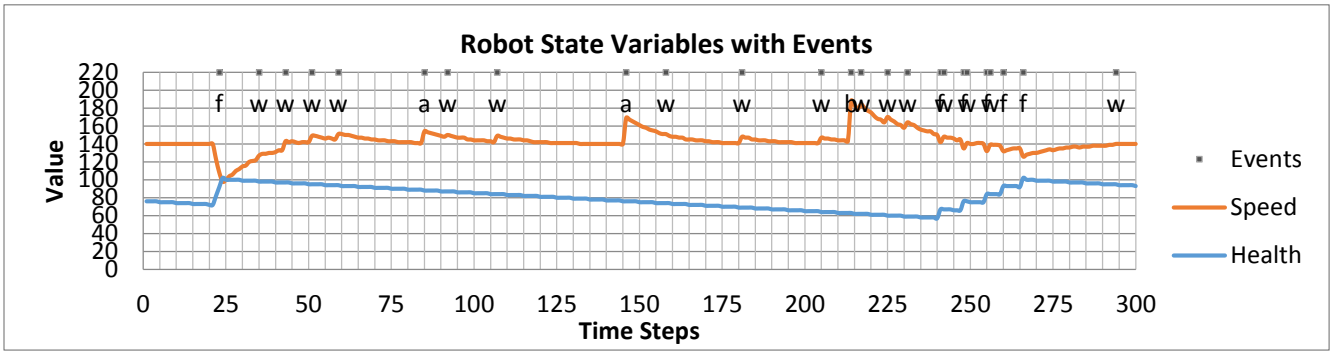


Figure 2: Shows the variation in the speed and health of the robot for a test scenario

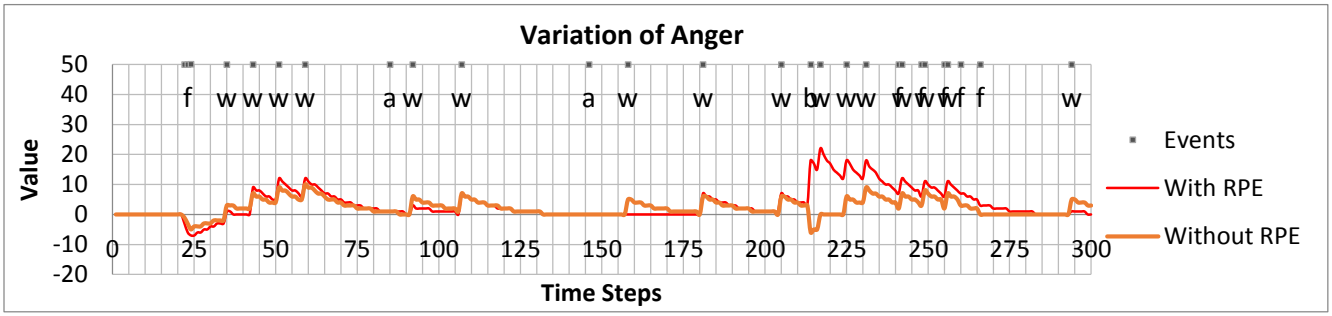


Figure 3: Compares the variation of anger with and without the RPE module

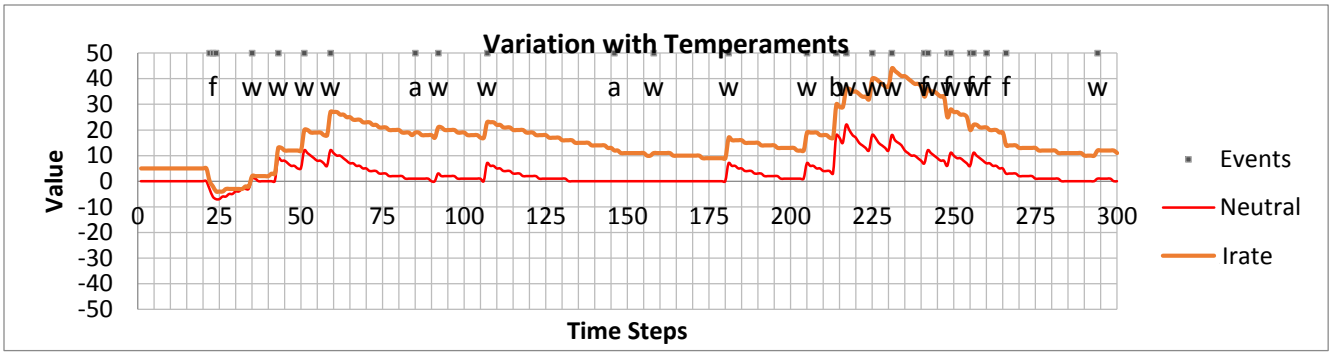


Figure 4: Compares the value of anger for an irate and a neutral temperament

Figure 4 shows the change in the value of the emotion anger for an irate temperament which was defined by modifying the weighting factors affecting the emotion anger.

Irate: The weighting factors used for anger were:

Emotion	w_0	w_1	w_2	γ
Anger	5	2.3	0.15	0.98

We observe that the steady state value of the affected emotion is offset from 0 by the value of the weighting factor w_0. The factor w_1 controls the instantaneous effect of an event on the emotions. As γ is increased we see that an emotion takes longer to decay to its steady state value. Thus an irate temperament causes the robot to have a higher steady state value of anger and remain angry for longer once an event angers it.

Conclusions

By adapting a model for momentary well-being (Rutledge et al., 2014) as done by Long et al. (2015), and incorporating a reward prediction error, an action selection mechanism with an integrated emotions engine has been implemented in addition to the development of a low cost robot to test the mechanism (Surendan, 2015). The ASM developed was a hybrid of a goal-based and a dynamic planning action selection mechanism. Each goal was associated with subgoals, and each subgoal with a dynamic plan. Different events were defined that affected the individual emotions and the ASM provided means by which the emotional state could be used to modify the internal state of the robot through means of command modifiers. Command modifiers also allowed the modification of dynamic plans and the value of emotions to be coupled through suitable models. Temperaments and emotional variability were defined using a matrix of constants. Varying the temperaments was observed to result in a different emotional state over the time period of the experiment even when the scenario was kept constant. Finally, the importance of the reward prediction error was highlighted by showing that without it events affected the emotions by the same amount regardless of when they occurred. With the RPE, it was possible to mimic the emotional response observed in humans by Rutledge et al (2014).

Additional tests should be conducted by specifying a more comprehensive list of command modifiers and fine tuning the temperament values based on the observation of an animal foraging for food in the wild. Since the temperament is specified by means of constants shown in Table 1, this leads to the possibility of an agent with a time-variant temperament that can alter its emotional sensitivity during run time. The emotional model could also be modified to introduce a time lag between the occurrence of an event and it affecting the emotional state. This would allow the simulation of the four classic temperaments, namely, melancholic, phlegmatic, choleric, and sanguine theorized by Greek philosophers ("Four Humors - And there's the humor of it: Shakespeare and the four humors," 2012). Another approach would be the specification of the temperament in terms of the "Big 5" traits of extraversion, agreeableness, openness, conscientiousness, and neuroticism often described in the literature (Digman, 1990). The low cost of the robot also permits the acquisition of a non-homogeneous robot swarm with varying temperaments to explore if emotions increase the effectiveness of the swarm.

References

Brom, C., & Bryson, J. (2006). *Action selection for Intelligent Systems*. European Network for the Advancement of Artificial Cognitive Systems.

Breazeal, C. & Brooks, R. (2004). Robot Emotion: A Functional Perspective. In *Who Needs Emotions,* J. Fellous (Ed.), Oxford University Press.

Digman, J. M. (1990). Personality Structure: Emergence of the Five-Factor Model. *Annual Review of Psychology*, *41*(1), 417–440.

Ekman, P. (1999). Basic emotions. In *Handbook of cognition and emotion*. Wiley & Sons Ltd.

Four Humors - And there's the humor of it: Shakespeare and the four humors. (2012, January 30). Retrieved from https://www.nlm.nih.gov/exhibition/shakespeare/fourhumors.html.

Jack, R. E., Garrod, O. G. B., & Schyns, P. G. (2015). Dynamic Facial Expressions of Emotion Transmit an Evolving Hierarchy of Signals over Time. *Current Biology*, *24*(2), 187–192.

Long, L. N. (2015). Modeling Emotion and Temperament on Cognitive Mobile Robots. *Presented at the 22nd Annual ACT-R Workshop*. Pittsburgh PA: Carnegie Mellon University.

Long, L. N., Kelley, T. D., & Avery, E. S. (2015). An Emotion and Temperament Model for Cognitive Mobile Robots. *Presented at the 24th Conference on Behavior Representation in Modeling and Simulation (BRIMS)*, Washington, DC.

Thompson, R. A., & Winer, A. C. (2015). The individual child: temperament, emotion, self, and personality. In *Developmental science: An advanced textbook, Edition: 6*. Psychology Press / Taylor & Francis.

Rutledge, R. B., Skandali, N., Dayan, P., & Dolan, R. J. (2014). A computational and neural model of momentary subjective well-being. *Proceedings of the National Academy of Sciences*, *111*(33), 12252–12257.

Picard, R. W. (2000). *Affective Computing*. Cambridge, MA: The MIT Press.

Kelley, T. D. (2006). Developing a psychologically inspired cognitive architecture for robotic control: The symbolic and subsymbolic robotic intelligence control system (SS-RICS). *Advanced Robotic Systems*, *3*(3), 219–222.

Surendran, V. (2015). *A dynamic action selection mechanism for an autonomous agent with a model for its emotional state*. M.S. thesis, Aerospace Engineering, The Pennsylvania State University, State College, PA.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Effects of Guanfacine and Phenylephrine on a Spiking Neuron Model of Working Memory

Peter Duggins (psipeter@gmail.com)
Terrence C. Stewart (tcstewar@uwaterloo.ca)
Xuan Choo (fchoo@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo
200 University Avenue West, Waterloo, ON, N2L 3G1, Canada

Abstract

We utilize a spiking neural network model of working memory (WM) capable of performing the spatial delayed response task (DRT) to investigate the functional effects of two drugs that affect WM: guanfacine (GFC); and phenylephrine (PHE). In this model, the loss of information over time results from changes in the spiking neural activity due to recurrent connections. We reproduce the standard forgetting curve, then show that this curve changes in the presence of GFC and PHE, whose application is simulated by manipulating various neuron properties. In particular, applying GFC causes increased firing in neurons that are sensitive to the information currently being remembered, while applying PHE leads to decreased firing in these same neurons. Interestingly, these memory-specific effects emerge from network-level interactions, because GFC and PHE affect all neurons equally. We compare our model to both electrophysiological data from neurons in monkey dorsolateral prefrontal cortex and to behavioral evidence from monkeys performing the DRT.

Keywords: working memory; delayed response task; guanfacine; phenylephine; Neural Engineering Framework

Introduction

Working memory (WM) is a central component of cognitive systems which use it to temporarily store information during the execution of complex tasks. Models of WM differ greatly between contemporary cognitive architectures, leading to diverse predictions about how information is represented and altered over time. Because WM is biologically realized in networks of neurons, one goal for researchers studying WM is to understand how networks of spiking neurons implement information storage and retrieval in the brain. In so doing, such models can be used to characterize deficits of WM associated with mental disorders, such as attention deficit hyperactivity disorder (ADHD) and Tourettes syndrome (Scahill et al., 2014), and be used to understand the biochemical mechanisms behind drugs used to treat such deficits (Avery, Franowicz, Studholme, van Dyck, & Arnsten, 2000). Due to the complexity of the interactions involved, few studies have characterized the relationships between drug chemistry, neurobiology, and cognitive abilities, including working memory.

In this paper we present a spiking neural network model of WM and action selection applied to a mnemonic cognitive test, the delayed response task (DRT). Computational models are well-suited to investigate multilevel interactions, including those between drugs that alter the brain's biochemistry and the resulting disruptions in cognitive abilities. We construct such a model using the Neural Engineering Framework (NEF) (Eliasmith & Anderson, 2003), a general method for building cognitive models from spiking neurons. The NEF has previously been used to create biologically-constrained models of list memory (Choo & Eliasmith, 2010) and action selection (Stewart, Choo, & Eliasmith, 2010) that are consistent with neural and behavioral data. This paper extends these models by simulating the effects of two drugs, guanfacine (GFC) and phenylephrine (PHE), which enhance and inhibit WM respectively.

In the next sections we describe the biological and computational basis of WM in the brain, examine the biophysical mechanisms of the applied drugs on neural activity, and advance a hypothesis for the relationship between them. We then present our model, describing how information is stored, forgotten, and retrieved in the delayed response task. When GFC (PHE) is applied to the model, we observe a shifted firing rate in those neurons whose spatial mnemonic tuning (preferred space/time direction) is aligned with the cue's location. This in turn affects the value stored in WM, leading to an increase (decrease) in performance on the DRT. The magnitude and timing of this effect is comparable to empirical data from monkeys. We conclude by proposing biophysical and anatomical extensions of the model.

Biological Background

WM is realized in the prefrontal cortex (PFC), a brain region whose prominent size in highly-evolved primates suggests its importance in complex cognitive tasks that require a flexible mental workspace. The PFC represents information that is temporarily held in mind and used to guide behavior and decision-making, and is thought to be maintained through recurrent excitatory connections between neurons with similar tuning properties (Goldman-Rakic, 1995). Computationally, this recurrence realizes an extended temporal integration that preserves the represented item without external stimulation (Singh & Eliasmith, 2006).

The stable representation of items stored in WM is particularly sensitive to the synaptic connections of intra-PFC loops and the biochemical environment of PFC neurons. Drugs that are used to treat WM disorders such as ADHD and Tourette's Syndrome target these biophysical mechanisms and have been shown to affect WM in healthy animals (Avery et al., 2000; Scahill et al., 2014). For example, guanfacine (GFC), an agonist for the α2A-adrenoreceptor, influences

WM in PFC neurons expressing Hyperpolarization-activated Cyclic Nucleotide-gated (HCN) ion channels (Franowicz et al., 2002). At rest, HCN channels permit the influx of non-specific cations, but deactivate in response to depolarization. HCN channels are unevenly distributed along the dendritic tree, with an almost sevenfold increase in density from the somatic to distal end of the dendrites. These properties allow HCN channels to reduce the temporal variability of dendritic excitatory postsynaptic potentials (EPSPs) that exists due to spatial distribution along the dendritic tree (Magee, 1999). It is believed that these channels control the excitability of pyramidal neurons in PFC by modulating the temporal dendritic summation and resting potential (Poolos, Migliore, & Johnston, 2002).

GFC, acting through a cAMP-mediated intracellular signalling cascade, closes HCN channels, resulting in less damping of excitatory dendritic spikes and increasing the overall excitability of the neuron. A study by Wang et al. (2007) showed that GFC increased the firing rate of PFC neurons with weak mnemonic tuning in the direction of spatial cues on the delayed response task, while having no effect on cells tuned in the opposite direction. Similarly, the α2A-adrenoreceptor antagonist PHE opened HCN channels and decreased firing rates of preferred-direction cells. These results are consistent with increased (decreased) behavioral performance on the delay-response task (Mao, Arnsten, & Li, 1999; Ramos, Stark, Verduzco, van Dyck, & Arnsten, 2006). We hypothesize that GFC raises the firing rate of spatially tuned neurons, causes a slower decay of items stored in PFC neural integrators, induces lower rates of forgetting, and consequently increases performance on the delay-response task. We test this hypothesis in a spiking neuron model.

A Spiking Neuron Model of Working Memory

The core requirement in a neural model of WM is a population of neurons that can maintain its state over time. That is, given a brief input, the internal connectivity should cause the neural activity pattern that results from that input to persist after the input has stopped. This persistence will not be perfect – over time the neural activity will drift away from its initial value.

However, this population of neurons cannot maintain *any* possible pattern of firing: we expect there to be correlations in the structure of this neural activity. Indeed, it has become common to analyze neural activity in WM areas (and elsewhere in the brain) by performing dimensionality reduction through techniques such as jPCA (Shenoy, Sahani, & Churchland, 2013). These approaches characterize the underlying patterns of correlation between the spiking neurons, identifying a lower-dimensional subspace that the neural activity represents. That is, rather than treating each neuron independently, we assume there is some vector x that is being represented by the population of neurons. The dimensionality of this vector is much smaller than the number of neurons, which means the information is redundantly encoded across these neurons. In particular, each neuron i will have some particular vector e_i for which that neuron fires most strongly (these are often known as "preferred direction vectors" or "encoders" and have been widely used as a useful way of characterizing cortical activity (e.g. Georgopoulos, Kalaska, Caminiti, and Massey (1982)). We can consider the total overall current going into a neuron to be proportional to $e_i \cdot x$ (the similarity between x and the preferred vector e_i). To produce a variety of tuning curves and firing rates that matches those in PFC, we randomly chose a gain α_i and bias current β_i for each neuron, resulting in a total input current of $\alpha_i e_i \cdot x + \beta_i$. This current can be fed into any neuron model, but here we simply use the standard leaky integrate-and-fire (LIF) model.

Given that the neural spiking activity encodes some vector x, it should be possible to recover that information by observing the spikes. The simplest method is to "decode" this spiking information via a weighted sum of the spikes, such that $\hat{x}(t) = \sum_i a_i(t) d_i * h(t)$, where $a_i(t)$ is the spiking activity of the ith neuron, $h(t)$ is the shape of the post-synaptic current[1] caused by the spikes, and d_i is the weighting factor for each neuron. The decoder (i.e., d_i) values can be found by performing a least-squares optimization that minimizes the difference between x (the original vector) and $\hat{x}$ (the vector recovered by observing the spiking activity). This method of characterizing neural representation is the first principle of the Neural Engineering Framework (NEF) (Eliasmith & Anderson, 2003).

Now that we have defined how a population of neurons can represent a value x, we can construct recurrent connections within this population such that the neural activity continues to represent x over time. To realize such a WM, we must find recurrent connection weights that stabilize dynamical neural activity, regardless of the value x being represented. Using the third principle of the NEF, this can be characterized as another least-squares minimization problem: previous work has shown that the optimal weights from neuron i to neuron j are $w_{ij} = \alpha_j e_j \bigotimes d_i$ (Eliasmith & Anderson, 2003). The result is a population of spiking neurons that maintains its activity over time, and has been the basis of multiple WM models (Singh & Eliasmith, 2006; Choo & Eliasmith, 2010).

To simulate the WM component of the delayed response task, we let x be two-dimensional, where the first dimension is the value to be remembered, and the second dimension is the amount of time it has been remembered for. Empirical and modeling evidence are consistent with the claim that PFC neurons explicitly encode the passage of time (Lewis & Miall, 2006; Bekolay, Laubach, & Eliasmith, 2014; Singh & Eliasmith, 2006). For example, some PFC neurons start firing only after a given amount of time has passed, while others gradually decrease their firing rate over time (Romo, Brody, Hernández, & Lemus, 1999). These "positive monotonic" and "negative monotonic" neurons can be thought of as neurons that are sensitive to both the value being represented and

[1] For this model, we use an exponential synapse with a decay constant of 100 *ms* consistent with NMDA-type glutamate receptors.

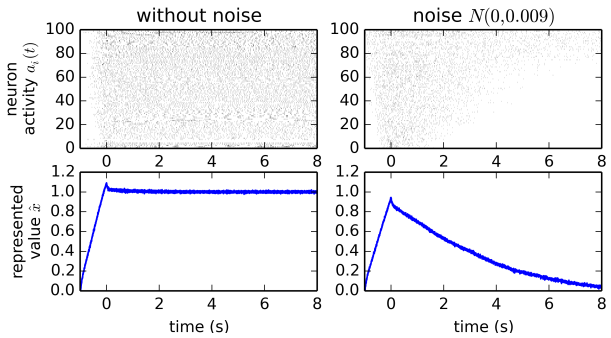


Figure 1: Recurrent spiking of WM neurons with and without added noise. Top spike rasters show 100 neurons (out of 3000). The represented value is computed from the spiking activity with $\hat{x} = \sum_i a_i(t) d_i * h(t)$. With random injected noise the memory is more unstable and decays towards zero.

the amount of time the memory has been held; in other words, these are *spatial mnemonic* neurons whose e_i values are large for both the first and second dimension. Other neurons may only be sensitive to one or the other dimension (i.e. would have small e_i values for one of those two dimensions). This variability in e_i matches well to the observed variability in WM tuning curves (Singh & Eliasmith, 2006).

Variability and Drug Effects

The WM model used here is based on that in Singh and Eliasmith (2006), with the addition of randomly varying background current to each neuron, to reflect the stochastic variability found in the brain. Without this random "noise", the information stored in WM is stable for a very long time (minutes to hours). However, with a small amount of background current added, the memory decays over tens of seconds as shown in Figure 1, consistent with decay rates of human WM (Choo & Eliasmith, 2010)

We use this model to investigate how WM is affected by the drugs GFC, which increases the excitability of neurons, and PHE, which decreases excitability. We simulate their effects using two alternative methods which simplify the aforementioned biophysics while maintaining the core functional properties in the NEF. In the first method, we model excitability as a global increase (or decrease) in somatic current to all WM neurons. Importantly, even though Wang et al. (2007) showed that, *in vivo*, an increase in firing activity was only observed for neurons whose preferred direction was aligned with the stimulus being remembered, we *do not* apply this extra current only to those neurons. This is because there is no direct mechanism by which GFC or PHE could affect only those neurons that are actively encoding information. Rather, we apply the simulated drug effect to *all* the neurons in the WM model. While this seems counter-intuitive, we show below that when we simulate this system, the network effects of the recurrent connections are sufficient to cause the observed differential response (Mao et al., 1999).

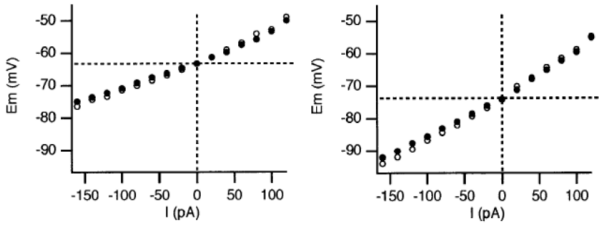


Figure 2: Subtheshold resting membrane potential as a function of applied current for normal (left) and HCN-knockout mice (right). Closing HCN channels lowers the neuron's resting potential (lower value of E_m at $I = 0$) while increasing the neuron's response to subsequent input (higher slope of E_m vs. I). Image reproduced from Nolan et al. (2004).

In the second method, we attempt to more faithfully reproduce the biophysical effects of HCN channel closure by manipulating the neurons' internal properties. HCN channels allow positive ions to flow into the cell, so closing HCN effectively induces a negative current, lowering the resting membrane potential. We model this effect by lowering the bias current β_i of each neuron in the WM. Additionally, closing HCN channels modulates neurons' dendritic summation, such that small, desynchronized dendritic spikes more strongly influence the somatic membrane potential. This effectively increases neurons' response to a given synaptic input, which we model by increasing the gain α_i of each neuron. We calibrate the competing effects of these manipulations using data from Nolan et al. (2004), which compares the subthreshold voltages of HCN-knockout mice and normal mice as a function of input current, Figure 2.

Modeling the Delayed Response Task

In the spatial delayed-response task (DRT), monkeys are presented with a brief (1s) visual cue positioned relative to their fixed gaze. The cue is removed. During the delay period (2, 4, 6, or 8s), the monkey stores the cue location in WM, then recalls that location in the response period by pressing the corresponding button or making a saccade. In terms of our model, the cue is considered to be a numerical value between -1 and 1 (the first dimension of the vector x). This value is fed into the model by directly injecting current into the WM neurons, causing them to spike with frequency determined by the similarity between their preferred vector e_i and the represented value x, computed as $e_i \cdot x$. This external current is injected for the duration of the cue period (1s) then removed; after this, the memory must be maintained by activity fed back through the WM recurrent connections.

To produce a response, the model must access that stored value and produce one of two outputs (-1 or +1). While a mechanism to perform this is straightforward to design with the NEF (Sharma, Kromer, Stewart, & Eliasmith, 2016), this part of the model does not alter the drug effects, so for simplicity we do not consider it here. Instead, we take the neural activity of the WM neurons and compute their weighted sum,

giving an estimate of the original value ($\hat{x}(t) = \sum_i a_i(t)d_i * h(t)$). Since a neural mechanism to convert this value into a decision will include some degree of variability, we approximate this by adding normally distributed noise to this value. If the result is above zero we interpret this as the model giving the first response, and if it is below zero we interpret it as giving the second response.

Results

To simulate the cellular effects of GFC and PHE, we tested two methods for perturbing the neurons, as described above. In the first, we injected a noisy signal[2] into neurons in the WM population, essentially using additive bias to increase (decrease) neural excitability. In the second, we manipulated the gains and biases of the LIF neurons used in the simulation, effectively decreasing each neuron's resting potential while increasing its gain to synaptic inputs[3]. Both perturbations produced the desired effects; we hereafter report results from the first method.

We began by comparing delay-related neural firing rates in the WM population[4] with activity from neurons in monkey dorsolateral PFC (Wang et al., 2007). We selected model neurons that were tuned to the preferred direction during control conditions, as per their hypothesized importance in representing the cue's location during the delay period. Wang et al failed to provide a precise definition of "weak spatial mnemonic tuning" or their procedure for choosing such neurons, so we selected model neurons based on the magnitude of their encoders (e), the change in their firing rate when presented with preferred-direction stimuli (da/dt), and their differential response to drug application (Δa). Figures 3 and 4 show the normalized firing rate of neurons before and after the simulated application of GFC and PHE. Both empirically and in simulation, GFC increased the firing rate of preferred-direction neurons while having little effect on neurons in the nonpreferred direction. Similarly, PHE decreased the firing rate of preferred, but not nonpreferred, neurons.

Next, we investigated whether the firing rate of preferred-direction neurons encoded the location of the cue stored in WM. Using the NEF, we decoded, from the neural activities, the value stored in the WM during the delay period. As the model forgot the original stimulus, this value decayed exponentially. In response to GFC (PHE), and concurrent with the increased (decreased) firing rate of preferred neurons, the WM value decayed less (more) rapidly, Figure 5.

Lastly, we tested whether the value stored in the WM coincided with the accuracy of the model on the DRT. Figure 6 shows the likelihood of correct response as a function of delay period length for a one-dimensional DRT (left-right cues). Both for monkeys (solid line) and the model (dashed line),

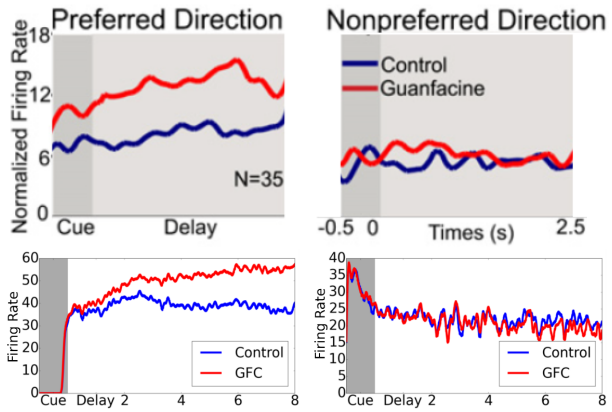


Figure 3: Normalized firing rate of neurons with spatial mnemonic tuning in response to Guanfacine. *Top*: data from monkey dlPFC while performing the DRT (Wang et al., 2007). *Bottom*: spikes from model neurons in the integrator population smoothed using Gaussian convolution ($\sigma = 0.04$ every $t = 0.2s$). GFC only increases the response of neurons that encode mnemonic information in the preferred direction. Preferred direction neurons: $0.3 < e < 0.7$, $0 < da/dt < 0.5$, $7 < \Delta a < 50$, $N = 25$. Nonpreferred direction neurons: $-0.7 < e < -0.3$, $-0.5 < da/dt < 0.5$ $-1 < \Delta a < 1$, $N = 5$.

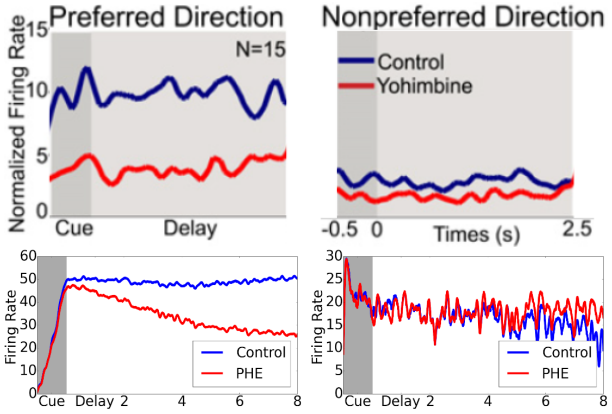


Figure 4: Normalized firing rate of neurons with spatial mnemonic tuning in response to α2A-adrenoreceptor antagonists yohimbine (*top*) (Wang et al., 2007) and phenylephrine (*bottom*). These drugs decrease the response of neurons with encoders in the preferred direction. Preferred direction neurons: $-0.7 < e < -0.3$, $0 < da/dt < 0.5$, $-50 < \Delta a < -10$, $N = 17$. Nonpreferred direction neurons: $0.3 < e < 0.7$, $-1 < da/dt < 1$ $-2 < \Delta a < 2$, $N = 2$.

[2]Normally distributed and proportional to the maximum firing rate, $N(0.002, 0.09)$ for GFC, $N(-0.002, 0.09)$ for PHE

[3]GFC: $\beta_{i,pre} = 0$, $\beta_{i,post} = -0.04$, $\alpha_{i,pre} = 1.00$, $\alpha_{i,post} = 1.036$; PHE: $\beta_{i,pre} = 0$, $\beta_{i,post} = 0.046$, $\alpha_{i,pre} = 1.00$, $\alpha_{i,post} = 0.960$

[4]$N = 3000$ neurons, neuron noise $\sigma = 0.009$, synaptic time constant $\tau_{=}0.1$, dimension $D = 2$ (stimulus, time).

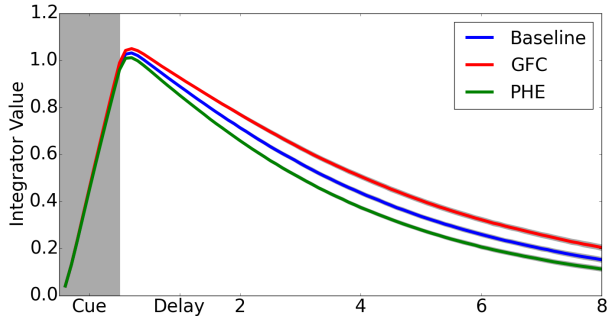


Figure 5: Value stored in WM during the delay period. Applying GFC (PHE) results in higher (lower) neural firing rates that shifts the curve up (down), altering the model's ability to distinguish the represented value from zero following the delay period. Reported values are averaged over $N = 50$ realizations with confidence intervals plotted in gray.

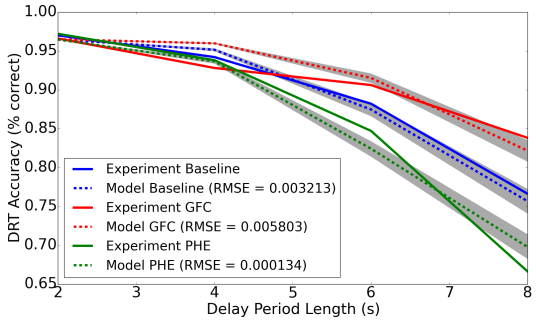


Figure 6: Accuracy on the DRT as a function of delay period length. Consistent with empirical results from monkeys performing the DRT, GFC increases accuracy while PHE decreases it. The outlier datapoint, experimental GFC at $t = 4$, probably arises from the small samplesize of the dataset: a single (unique) monkey was used for each experimental condition. *RMSE*: root mean square error between the empirical and model data. Reported values are averaged over $N = 50$ realizations with confidence intervals plotted in gray. Decision noise was fixed at $\sigma_{decide} = 0.19$ for all simulations.

accuracy decreased steadily from 2-6s then dropped sharply at 8s[5]. Our model shows increased (decreased) performance on the DRT following application of these drugs that fits the baseline empirical data with root mean-squared error between $0.0001 - 0.005$. The accuracy dropoff occurs when the value stored in the WM become indistinguishable from zero to the model's noisy decision procedure following the delay period.

Discussion

In this paper, we presented a minimal model of WM applied to the DRT that reproduces neural spiking and behavioral results under various drug manipulations with surprising accuracy. The model extends classical works on WM dynamics and the effects of neuromodulation (Brunel & Wang, 2001) by (a) incorporating the NEF, an approach that allows for the principled decoding of information represented in large-scale spiking neural networks, and (b) demonstrating that neuromodulation of WM (and its behavioral effects) can be studied through simple manipulation of neuron properties, bypassing the need to build complex circuits using Hodgkin-Huxley-type neurons.

Future work will address several simplifying assumptions made in the study. First, a detailed sensitivity analysis would reveal the robustness of the model to parameter variation. Exploratory experiments showed the decision noise and synaptic time constant altered the shape of the recall curves and increased the RMSE, but a more systematic investigation is needed to discover interactions between the remaining free parameters.

Second, the use of LIF point neurons to represent delay-related activity in WM necessitated an approximation of HCN opening and closing. Surprisingly, we found that both simple manipulations (biasing neurons or increasing their gain) produced changes in firing rate and behavioral response that match the empirical data. This suggests that biophysical simulations of drug-neuron interactions may be unnecessary, so long as the qualitative effect of the drugs on firing rate can be discerned from electrophysiological data. That said, replacing LIF neurons with more detailed neurons that include explicit HCN channels (which can be closed or opened by GFC or PHE) would expand the range of biochemical processes we could simulate. To progress in this direction, we have integrated the NEURON simulation package with the NEF-style modeling performed here.

Finally, while our model focused on the representational aspects of WM, the processes by which information is placed in, and retrieved from, WM are equally important for its implementation in unified cognitive systems. Adding input and output neural subsystems to the model, such as a visual hierarchy and a basal ganglia, would greatly expand the range of cognitive tasks that our model could potentially perform, avoid the use of arbitrary parameters such as the "misperception probability" and "decision noise", and present new targets for drugs that affect different aspects of cognition.[6] Many of these systems have already been built using the NEF (Eliasmith et al., 2012). In future work, we plan to implement both these extensions in pursuit of a deeper understanding of the neural basis, psychological dysfunction, and pharmaceutical modulation of working memory.

[5] To capture the inaccuracy of monkeys after a 2s delay, we introduced a 7% chance of misperceiving/ignoring the stimulus.

[6] For example, dopamine (D1) receptors are present both in PFC and hippocampus, and abnormal neurotransmitter/receptor levels have been implicated in WM deficits related to Parkinson's and schizophrenia (including performance on the spatial DRT).

Acknowledgments

This work was supported by CFI and OIT infrastructure funding as well as the Canada Research Chairs program, NSERC Discovery grant 261453, ONR grant N000141310419, AFOSR grant FA8655-13-1-3084 and OGS graduate funding.

References

Avery, R. A., Franowicz, J. S., Studholme, C., van Dyck, C. H., & Arnsten, A. F. (2000). The alpha-2a-adrenoceptor agonist, guanfacine, increases regional cerebral blood flow in dorsolateral prefrontal cortex of monkeys performing a spatial working memory task. *Neuropsychopharmacology*, *23*(3), 240–249.

Bekolay, T., Laubach, M., & Eliasmith, C. (2014). A spiking neural integrator model of the adaptive control of action by the medial prefrontal cortex. *The Journal of Neuroscience*, *34*(5), 1892–1902.

Brunel, N., & Wang, X.-J. (2001). Effects of neuromodulation in a cortical network model of object working memory dominated by recurrent inhibition. *Journal of computational neuroscience*, *11*(1), 63–85.

Choo, F.-X., & Eliasmith, C. (2010). A spiking neuron model of serial-order recall. In *Proceedings of the 32nd annual conference of the cognitive science society* (pp. 2188–93).

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A large-scale model of the functioning brain. *science*, *338*(6111), 1202–1205.

Franowicz, J. S., Kessler, L. E., Borja, C. M. D., Kobilka, B. K., Limbird, L. E., & Arnsten, A. F. (2002). Mutation of the α2a-adrenoceptor impairs working memory performance and annuls cognitive enhancement by guanfacine. *The Journal of neuroscience*, *22*(19), 8771–8777.

Georgopoulos, A. P., Kalaska, J. F., Caminiti, R., & Massey, J. T. (1982). On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *The Journal of Neuroscience*, *2*(11), 1527–1537.

Goldman-Rakic, P. (1995). Cellular basis of working memory. *Neuron*, *14*(3), 477–485.

Lewis, P. A., & Miall, R. C. (2006). Remembering the time: a continuous clock. *Trends in cognitive sciences*, *10*(9), 401–406.

Magee, J. C. (1999). Dendritic ih normalizes temporal summation in hippocampal ca1 neurons. *Nature neuroscience*, *2*(6), 508–514.

Mao, Z.-M., Arnsten, A. F., & Li, B.-M. (1999). Local infusion of an α-1 adrenergic agonist into the prefrontal cortex impairs spatial working memory performance in monkeys. *Biological psychiatry*, *46*(9), 1259–1265.

Nolan, M. F., Malleret, G., Dudman, J. T., Buhl, D. L., Santoro, B., Gibbs, E., ... others (2004). A behavioral role for dendritic integration: Hcn1 channels constrain spatial memory and plasticity at inputs to distal dendrites of ca1 pyramidal neurons. *Cell*, *119*(5), 719–732.

Poolos, N. P., Migliore, M., & Johnston, D. (2002). Pharmacological upregulation of h-channels reduces the excitability of pyramidal neuron dendrites. *Nature neuroscience*, *5*(8), 767–774.

Ramos, B. P., Stark, D., Verduzco, L., van Dyck, C. H., & Arnsten, A. F. (2006). α2a-adrenoceptor stimulation improves prefrontal cortical regulation of behavior through inhibition of camp signaling in aging animals. *Learning & Memory*, *13*(6), 770–776.

Romo, R., Brody, C. D., Hernández, A., & Lemus, L. (1999). Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature*, *399*(6735), 470–473.

Scahill, L., Chappell, P. B., Kim, Y. S., Schultz, R. T., Katsovich, L., Shepherd, E., ... Leckman, J. F. (2014). A placebo-controlled study of guanfacine in the treatment of children with tic disorders and attention deficit hyperactivity disorder. *American Journal of Psychiatry*.

Sharma, S., Kromer, B., Stewart, T., & Eliasmith, C. (2016). A neural model of context dependent decision making in the prefrontal cortex. In *Proceedings of the 38th annual conference of the cognitive science society.*

Shenoy, K. V., Sahani, M., & Churchland, M. M. (2013). Cortical control of arm movements: a dynamical systems perspective. *Neuroscience*, *36*(1), 337.

Singh, R., & Eliasmith, C. (2006). Higher-dimensional neurons explain the tuning and dynamics of working memory cells. *The journal of neuroscience*, *26*(14), 3667–3678.

Stewart, T. C., Choo, X., & Eliasmith, C. (2010). Dynamic behaviour of a spiking model of action selection in the basal ganglia. In *Proceedings of the 10th international conference on cognitive modeling* (pp. 235–40).

Wang, M., Ramos, B. P., Paspalas, C. D., Shu, Y., Simen, A., Duque, A., ... others (2007). α2a-adrenoceptors strengthen working memory networks by inhibiting camphcn channel signaling in prefrontal cortex. *Cell*, *129*(2), 397–410.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Capturing the Effects of Moderate Fatigue on Driver Performance

Ehsan B. Khosroshahi (ehsanebk@drexel.edu)
Department of Computer Science, Drexel University, 3141 Chestnut St.
Philadelphia, PA 19104, United States

Dario D. Salvucci (salvucci@drexel.edu)
Department of Computer Science, Drexel University, 3141 Chestnut St.
Philadelphia, PA 19104, United States

Bella Z. Veksler (bellav717@gmail.com)
Oak Ridge Institute for Science and Education
Cognitive Models and Agents Branch, Air Force Research Laboratory, 2620 Q St.
Wright Patterson Air Force Base, OH 45433

Glenn Gunzelmann (glenn.gunzelmann@us.af.mil)
Cognitive Models and Agents Branch, Air Force Research Laboratory, 2620 Q St.
Wright Patterson Air Force Base, OH 45433

Abstract

There is no doubt that fatigue plays an important role in driver performance and potential crashes. One way to understand the effects of fatigue on driving performance that has not been rigorously explored is to use cognitive modeling as a performance predictor. In this paper, we integrate existing models of driving and fatigue to make a general model of driving under the influence of moderate fatigue, caused by repeated bouts of nighttime driving. Empirical studies on the effects of moderate fatigue have shown that it affects two measures of performance: steering variability and lane variability. Moderate fatigue also tends to have an effect on night-shift conditions but not on day-shift conditions due to circadian rhythms. We describe our integrated model of a moderately fatigued driver and we analyze the model's performance in the context of a recent study of driving under conditions of moderate fatigue.

Keywords: Driving; fatigue; computational model; ACT-R

Introduction

Fatigue has always been one of the main contributors to car crashes (National Transportation Safety Board, 1999). This has been a motivation for numerous studies to document the effects of fatigue on driving performance. Although there have been a few previous attempts to model human performance under the influence of fatigue (e.g. Gunzelmann et al., 2011), these efforts were generally concerned with high levels of fatigue that arise from multiple days of sleep deprivation, which is unusual in naturalistic settings, and where crashes become prevalent. In our work here, we aim to develop a model of drivers with moderate levels of fatigue caused by nighttime driving, where dangerous incidents may happen with increased frequency but are not inevitable.

In this study, we integrate an existing model of driving (Salvucci, 2006) with a fatigue mechanism (Walsh, Gunzelmann, & Van Dongen, 2014), both implemented in ACT-R cognitive theory and architecture (Anderson, 2007). Then we evaluate the capacity of the integrated model to predict driving performance under moderate levels of fatigue. Based on the previous studies (Forsman et al., 2013; Van Dongen & Belenky, 2010; Van Dongen, Belenky, & Vila, 2011; Van Dongen, Jackson & Belenky 2010), we discuss which metrics or combination of metrics would be the most sensitive to driver drowsiness (moderate levels of fatigue) and how they change based on the time awake. We look at the same metrics in the model and examine fatigue-related effects as compared to behavior in a simple reaction-time task (PVT: Psychomotor Vigilance Test). This research expands on a previous integration (Gunzelmann et al., 2011) by modifying the mechanisms to reflect the current version of the ACT-R theory, and by exploring more detailed performance metrics to assess the model's performance.

Model of Fatigue

The model of fatigue used here is based on the work of Walsh et al. (2014), which is derived from the *state instability hypothesis* (Doran et al., 2001). State instability characterizes a person's fatigue as the switching between sleep and awake states, which may fluctuate second by second and can eventually progress to a physiological sleep state; the state instability hypothesis accounts, in general terms, for changes in performance associated with fatigue, including delayed response times, false alarms, and non-responses (see Doran et al., 2001). To represent the state instability hypothesis, Gunzelmann et al. (2009) introduced "micro-lapses" into a computational model to account for changes in behavioral performance. The concept of micro-

lapses heavily relies on ACT-R's procedural memory system and its sub-symbolic properties. The procedural memory is represented as productions, which represent condition-action *if-then* rules. During each production cycle in ACT-R, the conditions of the productions are evaluated to identify one that matches the current state, which is executed (fired) and changes ACT-R's internal state and possibly the external world. In cases where more than one production's conditions match the state of the world, values associated with each production rule, called utility, are used to determine which rule to fire.

By manipulating the utility of the productions and the utility threshold, the system is able to produce micro-lapses: if the utility (U_i) of the selected production is less than the utility threshold (UT), a micro-lapse occurs. Micro-lapses are production cycles during which no rule actually fires. Changes in U_i and UT influence the probability of micro-lapses occurring. To control changes in these values associated with fatigue, a *biomathematical* model is used to reflect time awake and circadian rhythms, whereas a *time-on-task* model is used to impose vigilance-related changes in the model.

Biomathematical models are based on neurophysiological and behavioral changes associated with sleep loss (e.g. McCauley et al., 2013; Achermann, 2004; Borb & Achermann, 1999). These models posit a two-process theory of alertness, specifically using time awake (homeostatic process) and time of day (circadian process) to determine the alertness level for a particular point in time. Despite the fact that biomathematical models do not give any insight to the specific cognitive and other components involved in task performance (like the changes in reaction time with the motor module), the integration with a cognitive architecture has given promising results (see Gunzelmann et al., 2009; Walsh et al., 2014). Following Walsh et al. (2014), the fatigue mechanism in ACT-R uses the alertness prediction of the biomathematical model to control dynamic fluctuations in production utilities and the utility threshold. For this study a recently updated biomathematical model developed by McCauley et al. (2013) was used. Time-on-task in the model is based on an exponential performance decline as the amount of time that is spent on a task increases (Giambra & Quilter, 1987). Based on these biomathematical and time-on-task models, and using the time of day that the task is taking place, the utility manipulations in ACT-R are formulated as follows:

$$FU_i = U_i * FP + noise$$

$$FP = FPpercent * (1 - FPBMC * biomath) * (1 + mpTime)^{FPMC}$$

The $biomath$ parameter is derived from the alertness prediction of the biomathematical model (McCauley et al., 2013) based on the sleep schedule and the hour that the task is happening. The $mpTime$ parameter represents the time (in minutes) that has passed since the start of each task. $FPBMC$, $FPMC$, are constants that are computed by regression coefficients and relate time awake and time of the day (biomathematical model) and time on task (time-on-task model) to produce a utility attenuation that is further moderated by $FPpercent$ parameter. $FPpercent$ represent the accumulated effect of micro-lapses on the utility calculations. The initial setting is 1 and has no effect before any micro-lapses occur. When a micro-lapse occurs, $FPpercent$ is reduced by a decay parameter which reduces overall utility value and increases the likelihood of another micro-lapse.

As described by Gunzelmann et al. (2009), since any task delay will cause $FPpercent$ to quickly decay to the point that will be too low to fire any production, there is a counterbalancing effect that resets $FPpercent$ (akin to awakening the model). For this reason, after any stimulus presentation, $FPpercent$ is reset back to 1.

A compensatory mechanism is in place to counteract the diminishing utility of the productions. The fatigue mechanism also manipulates the utility threshold based on a similar equation used for manipulating the utility:

$$UT = UT_0 * (1 - UTBMC * biomath) * (1 + mpTime)^{UTMC}$$

where the $biomath$ parameter is derived from the alertness prediction of the biomathematical model the same way as it was computed in the utility calculation, UT_0 is an initial utility threshold parameter, and $UTBMC$ and $UTMC$ are again constants provided to the mechanism.

Micro-lapses last for the duration of an ACT-R cognitive cycle (approximately 50 ms, plus noise). When micro-lapses happen, the model skips firing any production regardless of the state of the world. Individual micro-lapses are responsible for small increases in reaction times, while longer sequences produce more dramatic breakdowns in performance. As a result, they can be considered as the core of our model's account of driver behavior under fatigue.

Model of Driver Behavior and Fatigue

The model of driver behavior (Salvucci, 2006) was developed using ACT-R, and had the required cognitive processes for lane-keeping, lane-changing, and passing other vehicles. This model is based on a control model of steering behavior (Salvucci & Gray, 2004) which explains how drivers encode two points on the road: a near point in the lane center immediately in front of the vehicle, and a far point (such as vanishing point on a straight road) that provides stability while steering. The control law within the driver model aims to keep the far point stable while keeping the near point stable and centered, and these three components serve well as a theoretical account of both lane-keeping and lane-changing.

The core of the model uses a loop of four ACT-R production rules that (1) encode the near point, (2) encode the far point, (3) update steering and acceleration based on the position of the near and far points, and (4) check the

vehicle's stability and repeat this loop. Based on the ACT-R theory's 50 ms firing time for a production rule, the driver model results in a control cycle that requires roughly 200 ms for these four steps.

We integrated this driver model with the fatigue mechanisms described earlier, simply by running the driver model in the new version of the ACT-R architecture modified by the fatigue mechanism. Because of this integration, micro-lapses can occur between the cycles of the driver model. Since micro-lapses are cognitive cycles with no production executed, the driver model with micro-lapses takes longer to complete the control loop, negatively affecting performance. Our model also resets the *FPpercent* parameter at the end of each loop to counterbalance the effects of decay caused by micro-lapses as discussed earlier. The skipped rule firings then are a reflection of the basic prediction that such micro-lapses represent fatigued behavior.

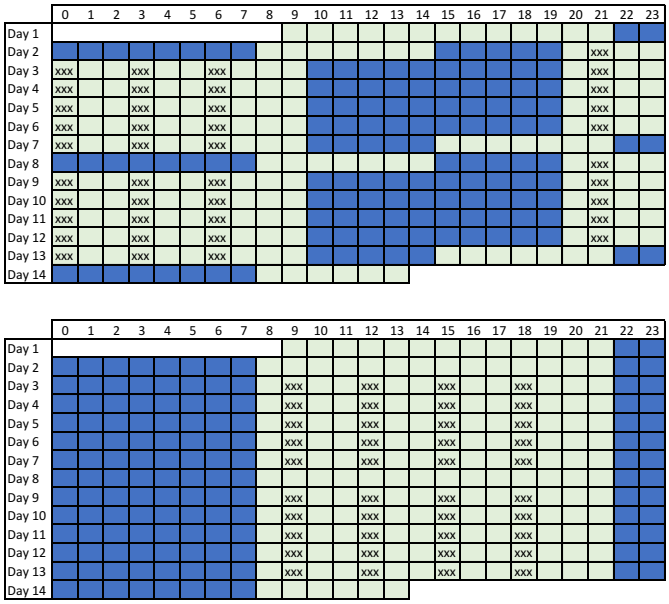


Figure 1: The experimental study protocol for the night shift (top) and day shift condition (bottom) in study A. Blue indicates the sleeping time. The three X's indicate the PVT/driving/PVT sessions. The protocol for study B was equivalent to the night shift of study A (top), except for one extra day at the beginning and one extra day in the break.

Experiment and Model Results

Forsman et al. (2013) used data from two laboratory experiments with the same driving scenario to measure the changes in driving performance metrics with the levels of driver drowsiness. The experiment included 41 participants in two studies (A and B): study A (Van Dongen & Belenky, 2010; Van Dongen et al., 2011) had a night and a day shift (14 days); study B (Van Dongen et al., 2010) had only the night shift (16 days). The night shift of study A and study B were basically equivalent, except for an extra baseline day (added between days 1 and 2) and an extra restart day (added between days 7 and 8). The study protocols are illustrated in Figure 1. Participants completed driving sessions at 21:00, 00:00, 03:00, and 06:00 for the night shift, and at 9:00, 12:00, 15:00, and 18:00 for the day shift (we refer to these session times as time points 1 to 4, respectively, for both night and day shifts). Every session included a 30-minute driving session with a 10-minute psychomotor vigilance test (PVT) before, and another following, the driving session.

Psychomotor Vigilance Test (PVT)

As mentioned, each driving session in the experiment was preceded and followed by a 10-minute Psychomotor Vigilance Test (PVT). The PVT is a simple reaction time task that can be an independent measure of fatigue (Van Dongen et al., 2011). The main dependent measure in the PVT was the number of lapses in the experiment which are reaction times longer than 500 ms. Only the pre-driving PVT lapses were used by Forsman et al. (2013) in their data analysis to have a measure of fatigue of participants just before the driving test. The experimental results for the PVT lapses based on the time points are shown in Figure 2. The result is based on the two 5-day shift period separated by 34-hour break, averaged on the time points. Clearly, the number of lapses increases as a function of the number of hours awake in the night shifts. This increase illustrates the use of PVT test as a measurement of fatigue.

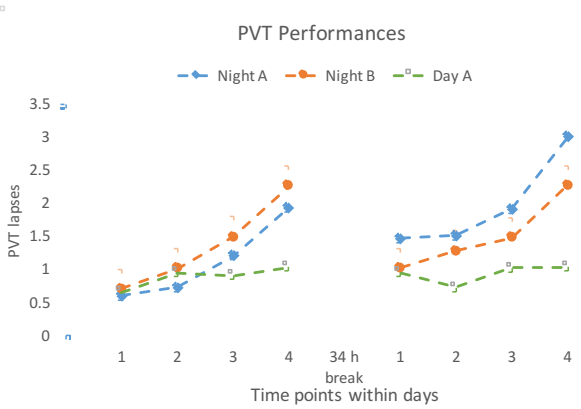


Figure 2: Experimental result for PVT lapses.

We developed a cognitive model of the 10-minute PVT test in Java ACT-R [http://cog.cs.drexel.edu/act-r/] using the fatigue model explained earlier. The schedule of the biomathematical model's sleep schedule was set to exactly match the experiment. At each session of the test, the model looks at the stimuli and responds as soon as possible. When using the fatigue mechanism, the response times reflect the number of micro-lapses that occurred during the firing of productions between seeing the stimuli and responding to it. The model was run 100 times and the corresponding results are shown in Figure 3. The numbers of lapses were

aggregated based on the time points used in the empirical studies.

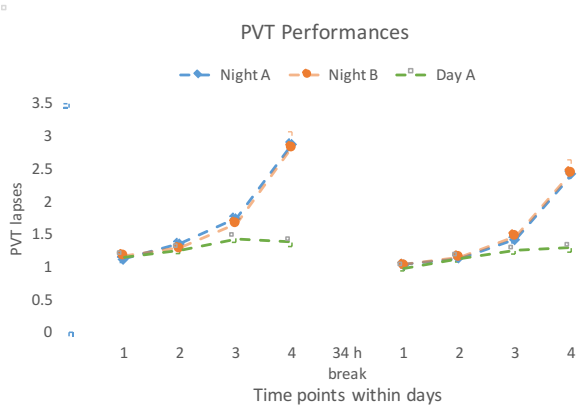


Figure 3: Model result for PVT lapses.

From the experimental data (Figure 2) we can clearly observe that that the PVT lapses increase during night shifts but not during day shifts. This fact led Forsman et al. to use the day shift in study A as the control condition (study B did not have a day shift). We see the same effect more strongly in the model's results (Figure 3). The correlation of the experimental data and model results also showed a significant correspondence ($r = 0.77$; $p < 0.001$), providing further validation for Gunzelmann et al. (2009)'s fatigue mechanisms and also giving us settings for the critical fatigue-related parameters used in the driving task described in the next section.

Driving Experiment

The driving scenario was in daylight with a clear view on a rural highway with no wind and without other cars. During each driving sessions, participants drove about 30 minutes on a 28-mile track. They were instructed to maintain their speed at 55 mph. Along the 28-mile track, there were ten half-mile straight segments during which the primary signals were captured. Forsman et al. (2013) extracted 87 driving metrics from the time series of the concatenated straight segments for each session.

When performing principal component analysis (PCA) on the values for these metrics, they discovered that there were two dominant dimensions in both studies (A & B). The first dimension, which explained the highest portion of both studies' total variance in the driving datasets, showed high factor loading for metrics capturing steering variability, which they referred to as the *steering variability* component. The second dimension, which explained the second-highest proportion of the variance, showed high factor loading for the metrics capturing the variability in lateral lane position, which they referred to as the *lane variability* component.[1] The results revealed that among the 87 metrics, these two components captured the largest portion of the total variance in the fatigued condition (47% for study A and 44% for study B).

Interestingly, they found that although the first dimension (*steering variability*) explained more variance than the second dimension (*lane variability*), the latter correlated more to the PVT performances at the time points within each day, which are critical as independent markers of fatigue in our data analysis here. Specifically, the proportion of steering wheel movements exceeding three degrees in angle ($STEX_3(S)$) had a very high factor loading in steering variability (highest in study A and fourth in study B), and standard deviation of lateral lane position ($SD(L)$) had the highest in both studies' lane variability. This led us to use these two metrics as representative of steering and lane variability in our model's data. Their result also showed between the metrics having high factor loading in the first two dimension, there were not any metrics capturing speed variability. The factor loading for the speed related metrics were not reported in the experiment, so we picked standard speed deviation ($SD(V)$) as a representative for speed variability.

Driving Model

These findings from the driving experiment led us to build a model of driving and examine its performance on metrics related to steering, lane, and speed on a similar experimental driving scenario. We developed a cognitive model using the driving model in Java ACT-R (Salvucci, 2006) and integrated the fatigue mechanism as mentioned before with the same sleep schedule as in the experiment. The only difference between the model's scenario and the experimental scenario was that we extracted driving metrics for 30 minutes of straight driving on the highway without any other cars, compared to the concatenated straight segments in the experiment. During every session, the model drove on a straight highway without changing lanes.
The estimated fatigue parameters for the driver model were taken directly from the earlier PVT model with no changes. The model was run 20 times and we averaged the driving metrics by time points. We particularly concentrated on extracting $SD(L)$, $STEX_3(S)$, and $SD(V)$ as defined above. The corresponding results for these variables are shown in Figures 4, 5, and 6. By looking at these figures, we can easily see changes in $SD(L)$, $STEX_3(S)$, and $SD(V)$ across time points. Comparing the PVT result, the model captured a slightly higher correlation of $SD(L)$ and $STEX_3(S)$ ($r = 0.954$ for both) compared to $SD(V)$ ($r = 0.937$). More interestingly, the correlation of these variables and PVT results for each study (Table 1), indicates that both $SD(L)$ and $STEX_3(S)$, correlate strongly on night-shift conditions whereas this relationship is much weaker in day-shift condition for $SD(L)$ and slightly weaker for $STEX_3(S)$. The

[1] Although steering and lane variability are certainly related, Forsman et al. (2013) argue that they are "statistically orthogonal" in their analysis because of the filtering effect of the vehicle's physical dynamics.

relationship of $SD(V)$ and PVT, showed a reverse result, meaning that day shift had a stronger relationship than the night shifts. By looking at Figures 4, 5, and 6, we can also see an upward trend across the time points (higher levels of these variables indicate worse performance) in the night shifts compared to the day shift, which has a much more moderate increase. This is an indicator of the moderate fatigue effects in driving performance as a function of time of day in night-shift conditions as compared to the day-shift condition.

Table 1: Correlations coefficient (r) between variables and independent indices of fatigue (PVT) in the model.

	Study A night shift	Study B night shift	Study A day shift
$SD(L)$ vs PVT	0.968	0.972	0.392
$STEX_3(S)$ vs PVT	0.980	0.972	0.917
$SD(V)$ vs PVT	0.928	0.964	0.977

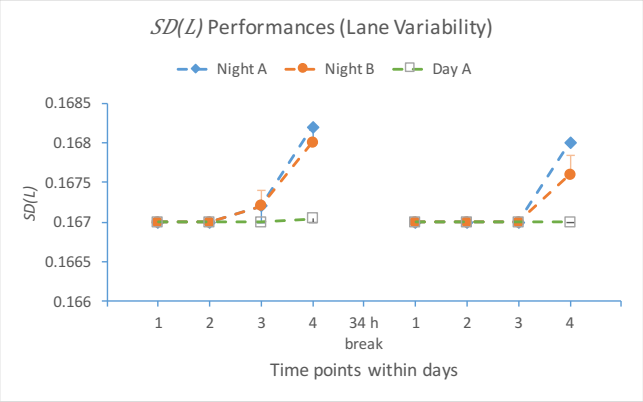


Figure 4: Model result for lane variability.

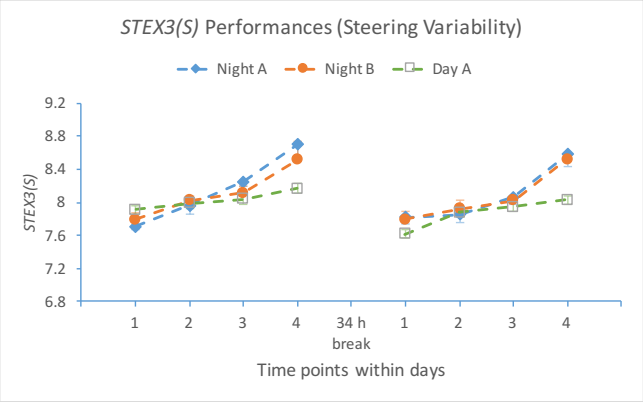


Figure 5: Model result for steering variability.

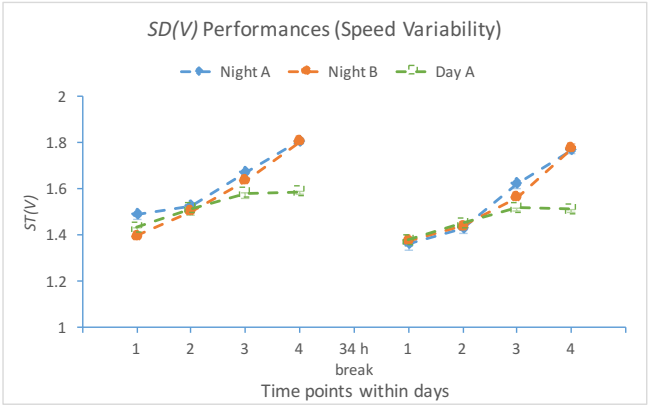


Figure 6: Model result for speed variability.

Discussion

One important way to understand and prevent driver error and crashes is to detect driver's drowsiness before the high level of fatigue makes crashes more likely. Previous work (Forsman et al., 2013) has revealed that among a large number of driving performance metrics, there were two independent components that capture significant variance that reflect the effects of moderate fatigue: steering-related metrics and lane-related metrics. Thus, accounting for changes in driver behavior related to these two metrics is a good way to understand driver performance and potential sources of problems.

In this paper, we developed a model of driving using a cognitive architecture and tested it with moderate levels of fatigue. Overall, the model captured the broad pattern of variance in steering metrics and lane position metrics based on the levels of fatigue. The model also captures the night-shift changes versus the day-shift. Thus, the model has shown promise in accounting for moderate levels of fatigue, extending previous results (Gunzelmann et al., 2011) by updating the model and mechanisms to the latest version of ACT-R and extending the earlier validation for more severe levels of fatigue. One of the more interesting parts of the model is the fact that the variables for the fatigue model were directly taken from the pre-driving PVT model. This shows the capacity of the integration in capturing different performances for different levels of fatigue. Ultimately, this is a nice step toward finding a general model of fatigue in driving, without free parameters and without the need for empirical driving studies that include precise measurements of fatigue. Such a model would then be applicable to challenges such as interface system design and evaluation (see Gunzelmann, Veksler, Walsh, & Gluck, 2015).

One of the main limitations and potential areas for future work on the driving model is the need of a better simulation of mechanics and dynamics of an actual car to predict fatigued performance for a particular driver and vehicle

pairing. For example, we faced problems in adapting the model to a driver trying drive at a constant speed without any automatic cruise controls. The acceleration pedal in the current model cannot capture the dynamics of car speed changes as well as in real cars, and assumes that the driver is in complete control of speed—whereas, with modern systems like automatic cruise control, fatigue may affect driver performance in very different ways than examined here. As models of driver behavior adapt to these new technologies, we expect that these initial steps in modeling fatigue will still largely generalize to more complex driving situations.

Acknowledgments

This work was funded in part by a grant from the Air Force Research Laboratory (#FA8650-15-2-6603).

References

Achermann, P. (2004). The two-process model of sleep regulation revisited. *Aviation, space, and environmental medicine*, *75*, A37-A43.

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press.

Borb, A. A., & Achermann, P. (1999). Sleep homeostasis and models of sleep regulation. *Journal of Biological Rhythms*, *14*, 559-570.

Doran, S. M., Van Dongen, H. P. A., & Dinges, D. F. (2001). Sustained attention performance during sleep deprivation: evidence of state instability. *Archives italiennes de biologie*, *139*, 253-267.

Forsman, P. M., Vila, B. J., Short, R. A., Mott, C. G., & Van Dongen, H. P. (2013). Efficient driver drowsiness detection at moderate levels of drowsiness. *Accident Analysis & Prevention*, *50*, 341-350.

Giambra, L. M., & Quilter, R. E. (1987). A two-term exponential functional description of the time course of sustained attention. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *29*, 635-643.

Gunzelmann, G., Gross, J. B., Gluck, K. A., & Dinges, D. F. (2009). Sleep deprivation and sustained attention performance: Integrating mathematical and cognitive modeling. *Cognitive Science*, *33*, 880-910.

Gunzelmann, G., Moore, L. R., Salvucci, D. D., & Gluck, K. A. (2011). Sleep loss and driver performance: Quantitative predictions with zero free parameters. *Cognitive Systems Research*, *12*, 154-163.

Gunzelmann, G., Veksler, B. Z., Walsh, M. M., & Gluck, K. A. (2015). Understanding and predicting the cognitive effects of sleep loss through simulation. *Translational Issues in Psychological Science*, *1*, 106-115.

McCauley, P., Kalachev, L. V., Mollicone, D. J., Banks, S., Dinges, D. F., & Van Dongen, H. P. (2013). Dynamic circadian modulation in a biomathematical model for the effects of sleep and sleep loss on waking neurobehavioral performance. *Sleep*, *36*, 1987-97.

National Transportation Safety Board (1999). Evaluation of U.S. Department of Transportation efforts in the 1990s to address operator fatigue. *National Transportation Safety Board*, Safety Report NTSB/SR-99/01, Washington, D.C.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors*, *48*, 362-380.

Salvucci, D. D., & Gray, R. (2004). A two-point visual control model of steering. *Perception*, *33*(10), 1233-1248.

Van Dongen, H., & Belenky, G. (2010). Investigation into motor carrier practices to achieve optimal commercial motor vehicle driver performance: Phase I (No. FMCSA-RRR-10-005).

Van Dongen, H. P., Belenky, G., & Vila, B. J. (2011). The efficacy of a restart break for recycling with optimal performance depends critically on circadian timing. *Sleep*, *34*, 917-929.

Van Dongen, H. P., Jackson, M. L., & Belenky, G. (2010). Duration Restart Period Needed to Recycle with Optimal Performance: Phase II. U.S. Department of Transportation, Federal Motor Carrier Safety Administration, Office of Analysis, Research and Technology.

Walsh, M. M., Gunzelmann, G., & Van Dongen, H. P. (2014). Comparing accounts of psychomotor vigilance impairment due to sleep loss. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society* (pp. 877-882).

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Using Naturalistic Typing to Update Architecture Typing Constants

Marc T. Burns (mtb137@psu.edu)
Frank E. Ritter (frank.ritter@psu.edu)
Xiaolong "Luke" Zhang (lzhang@ist.psu.edu)
College of Information Sciences and Technology, Penn State University
University Park, PA 16802 USA

Abstract

Despite the near-ubiquity of graphic user interfaces for navigating the digital and virtual space, relatively little is known about their naturalistic usage. We start to address two questions. First, how do people use computers outside of laboratory studies? This includes what we can we determine about user behavior by analyzing detailed user logs. Second, can we update the constants in user and cognitive models for predicting typing time based on naturalistic behavior? We thus recorded naturalistic logs of 45 users over 219 sessions providing 1,865 hours of behavior (average session=8.18 hours). The analyses of keystroke times are sensitive to the definition of typing (e.g., how close keys are to be counted as continuous typing), and comparisons will need to provide clear definitions or tradeoff curves. Using this data, we updated the typing and homing constants of the Keystroke-Level Model (Card, Moran, & Newell, 1983), a theory of interface behavior used to provide constants for many cognitive architectures. The results suggest that people are typing faster than previously believed based on the 1983 KLM predictions; homing (moving one's hand from mouse to keyboard and keyboard to mouse) occur frequently, and now appear to be different events and thus require separate constants.

Keywords: GOMS, KLM, Typing Rates

Introduction

How do people use keyboards when they are not working in the lab? To investigate this, we performed a naturalistic study of how users interact with their computers in their natural settings. Because users are spending increasing time interacting with their computers earlier in life, the need to understand how users interact is increasingly important.

This study differs from previous research by analyzing the results from users' natural computer usage. By studying users in a variety of settings where they use their own computers, in participants' offices and homes, we are able to log users' actual behaviors as they occur. This behavior differs from previous laboratory studies due to the naturalistic approach to this study.

Studies focusing on how users perform tasks in labs risk diminished authenticity of the data collected because users may be working on unfamiliar computers in an unfamiliar context (the laboratory setting) using an unfamiliar keyboard to do a new, prescribed task. For example, few if any studies in our experience ask participants to set the keyboard settings to their preference. This study addresses these issues through a naturalistic study of human-computer interaction (HCI).

Users' interaction with their computer interfaces is significant for several reasons. First, one can develop an understanding of users' inefficiencies and capabilities. Second, targeting users' fundamental inefficiencies and building from their capabilities can provide developers insight into designing easier and more efficient interfaces. It may be necessary to study users' natural interaction tendencies to predict accurate interface times.

Lastly, a remaining challenge facing the study of users, and more broadly HCI, is that of accurately mapping what users actually do, when they do it, and why they do it. Lab studies like Card et al. (1983) can help us gain accurate understanding of user interaction behaviors, but this method assigns tasks to users, rather than lets users organically select their own tasks. While their experimental design was highly effective in quantifying specific tasks, it is probably inaccurate to assume that modern computer users follow the same single tasked-ness (e.g., Salvucci & Taatgen, 2011; Spink, Ozmutlu, & Ozmutlu, 2002). Additionally, experiment-driven research cannot account for patterns of user task selection, task switching behavior, and users' breaks in natural computer use. For example, few experiments account for users watching videos during the completion of their task. Thus, to gain an accurate representation of how users act in their day-to-day usage, naturalistic research is required. Furthermore, little research has been completed since Card, et al. (1983) to update the constants in the KLM; these may have changed with increased usage and with a broader population of users.

Previous Studies and the KLM Model

We note here several studies that make suggestions about how we should proceed with this work.

Early Studies and the KLM

Kinkead (1975) investigated the differences in keyboard layouts, and found that there is a nominal difference across layouts, suggesting that the speed of typing is based more on how practiced a keyboardist is, rather than an "optimal" keyboard layout. Kinkead concluded that a vast majority, 95%, of keystrokes occur in 2/3 s, or 667 ms, which is much slower than the times defined later in Card et al. (1980). However, similar to Card el al., these users performed prescribed tasks.

In 1980, Card et al. published a paper outlining the Keystroke-Level Model (KLM), a simple tool to aid in the designing of interactive computer systems. The KLM used the time required to perform the sub-steps that make up a task on a computer to predict the time it takes an expert to perform said task (Card et al., 1980). The original purpose

of the KLM was a rough calculation for system design. However, in creating the KLM, Card et al. extensively studied the sub-tasks, including physical-motor operators that make up the task, as well as the mental priming required to perform the task.

Keystroking is the time required to strike one key. The KLM treats all keystrokes the same, regardless of the key. These times range from 0.4 s per character for the best typist to 1.20 s per character for the worst typist on an unfamiliar keyboard.

Homing time is defined as the time for a user to move their hand between devices. This motion is from the keyboard to the mouse, or vice-versa. Card et al. reported homing time as 0.4 s during a text-editing task.

Surprisingly, despite dramatic changes in the early adoption, frequency of use and format of computers, we find few recent studies in this area. Additionally, even at the time of Card et al., the rate-limiting step of computer users in text-editing tasks was not the interface of their computer, but rather the information-processing capacity of the user (Card & Moran, 1986).

The first challenge is to expand the applicability of the research by Card et al. (1980). To simplify the process of creating models that accurately describe computer user behavior, Card et al. solely studied the task of text editing under several strict restrictions:

- Users could only be expert computer users
- The task must be routine
- Performance must be without errors.

This contextual rigidity allowed Card et al. to develop the Keystroke-Level Model and GOMS (Goals, Operators, Methods, and Selection rules), two widely used human-information processor models. However, these models do not apply to all users, tasks, and metrics. Both the Keystroke-Level Model and GOMS focus on time required by experts to complete a task as the unit of measurement. These constants are used in ACT-R and EPIC, and sometimes with Soar models to check timing predictions. These models are unable to address other fundamental metrics such as the quality of work. They also did not study users doing tasks that users complete outside of lab experiments.

Thus, this paper focuses on the physical-motor tasks of typing and homing, and the time required to perform these tasks. Despite the KLM model allowing 1.35 s mental preparation at the beginning and ending of tasks, as well as system response time, which is dependent upon the system being used, our study examines what constitutes continuous typing, the time to perform a keystroke within continuous typing, what comprises of a homing, and the time to complete a home.

Card and Moran (1986)

Three years after the publication of *The Psychology of Human-Computer Interaction*, Card and Moran (1986) revisited some of the assumptions and building blocks upon which their book was written. They outlined four interfaces for modeling the interaction between humans and computers: physical, cognitive, conceptual, and task. These four interfaces constitute the foundation of the literature that this paper updates.

Later Updates

Since the studies of Card, Moran, and Newell, many researchers have moved to answer remaining questions, on how users interact with computers. Unfortunately, studies to answer these questions typically and nearly exclusively occur in carefully controlled, task-oriented contexts (MacKenzie, Sellen, & Buxton, 1991; Whisenand & Emurian, 1999).

To that end, we study participants in a naturalistic environment to obtain data more representative of actual behavior and to build a comprehensive representation of how GUIs are used in daily computer interactions.

Summary

Thus, the constants in the KLM theory have been used for a while and could be updated based on users' increased experience with interfaces. The data used in the KLM have been obtained from experimental studies, as opposed to taken from users performing their own tasks on their own devices in a naturalistic setting. To improve our understanding, we will record users' mouse movements and keystrokes while they perform their normal daily tasks on their own computers.

Method

Data was collected by recording the users' keystrokes and mouse movements while they performed their typical tasks at home and at work. We used an anonymizing keystroke logger for privacy.

With this data, we start to determine what constitutes typing, and the naturalistic typing speed; a rich area that we are just beginning to explore. We also provide an update to the typing constants from Card et al. (1983).

The input logger records mouse clicks, mouse movements, and keystrokes across all tasks. Furthermore, such information is useful for inferring patterns of natural behavior, such as how long participants use a computer at a session, total daily usage, number of keystrokes, typing speed, etc.

Participants encounter the logger only upon the start and conclusion of a session, without artificial tasks, distractors, or external observers or apparatus.

Participants

Of the 45 unpaid participants 18 were male, 12 female, and 15 declined to report. In addition, 3 participants were left-handed, 17 were right-handed, and 25 declined to report handedness; however, each participant used their preferred hand for using the mouse. Participants were members of the Penn State community, with a majority (20) being

undergraduate students in the College of IST, 16 employees including office staff and professors, and 9 graduate students. We assigned subjects IDs between 1 and 100.

Apparatus

Participants used their own hardware: 44 used Windows on PCs, while one used OS-X on Apple hardware.

We recorded the users' inputs with Recording User Input ver. 2.03 (RUI), a keystroke and mouse logging tool (Kukreja, Stevenson, & Ritter, 2006; Morgan, Cheng, Pike, & Ritter, 2013) that runs on Windows and Macintosh computers. This modified version of RUI anonymized most keystrokes. Individual keystrokes were replaced with the asterisk character; combination key keystrokes (chords), including shift, alt, and control preserved the special key being pressed with an asterisk to signify a keystroke.

Design and Procedure

Participants were provided a thumb drive with RUI or had RUI installed on their computer by an experimenter. RUI ran in the background while participants used their computers. Sessions typically lasted a workday (mean of 8.18 hours), ending when the participant terminated RUI and the thumb drive was returned to the experimenter. Participants typically recorded four sessions.

Keystrokes included as continuous typing in later analyses met the criteria in Table 1, while user actions included as homing in further analysis met the criteria defined in Table 2.

Table 1: Criteria for including keystrokes in continuous typing calculations.

- Contiguous (no intervening mouse events)
- Not the first of its action, e.g. must follow a keystroke
- Not an alt or control key chord
- Occur within 2 s of the previous keystroke

Table 2: Criteria for Homing.

- Mouse movement following a keystroke; or keystroke followed by a mouse movement
- An additional mouse movement occurs following the initial mouse movement of 10 or more pixels; or an additional keystroke must follow the initial keystroke
- Occurred in 2 s or less from previous action

Results and Discussions

Overall, our 45 participants logged 1,816 hours of interaction including over 1.5 million keystrokes. From the ~60.3 million records (over 3 GB), we computed derived measures that we present in two sections: (a) the naturalistic typing behaviors, (b) homing actions.

Analysis of Keystroke Data

What is continuous typing? The typing rate can be calculated as the total number of keys divided by time for a fixed amount of text. Analysis of the raw data showed an average typing speed over the whole sessions (including all pauses) of 13.7 characters per minute or 2.25 words per minute (wpm) assuming 82% of characters typed are word characters with a word of 5 characters (MacKenzie, 2002; Salthouse, 1984).

Because our data is naturalistic, it is not possible to know when the intention to type starts. So we needed to determine a cutoff for when typing begins and ends. In our analysis, we define typing as two or more contiguous keystrokes non-interrupted by mouse movements or button clicks. We computed and plotted the mean time between keystrokes. Figure 1 shows how the typing rate varies as the time threshold allowed between keystrokes varies from 1 ms to 30,000 ms across all users. At 0 ms, one cannot compute a typing rate, and 30 s represents a relatively long time between keystrokes.

In Figure 1, at around 10 s, the mean keystroke time starts to flatten. Ten seconds is still probably too large to consider being a dwell time between keystrokes in continuous typing. Card, Moran, and Newell (1983) in the KLM and Kieras in GOMS (Kieras, 1988) note 1.20 s per keystroke for an inexperienced typist on unfamiliar devices. We thus round up and use 2 s as a threshold in the remainder of this paper, although Figure 1 shows that the computation of keystroke time is sensitive to this threshold.

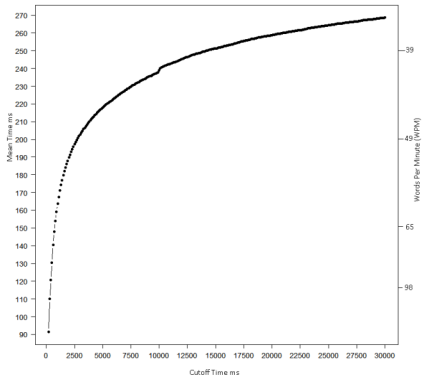


Figure 1: Mean time for keystroke (ms) vs. cutoff time (ms).

Figure 2a shows that the distribution of the times between keystrokes (with < 2 s separation) has a long tail. Figure 2b shows that individual distributions also have long tails. This effect is likely caused by distractions, such as leaving the keyboard, other task actions performed on other devices such as phones, or other events occurring while the user was typing or entering short strings.

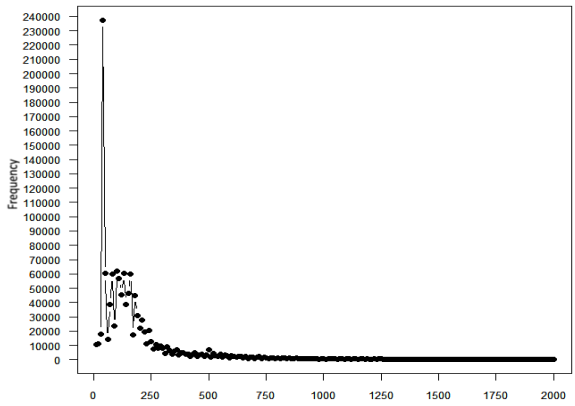


Figure 2a: Keystroke time distribution 2 s threshold in 2 ms bins across all users.

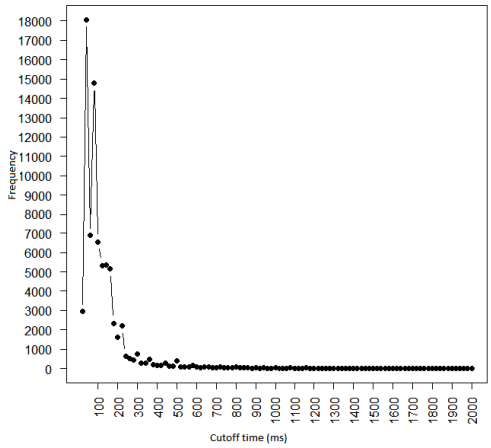


Figure 2b: Subject 6's keystroke distribution with 2 s cutoff in 20 ms bins.

Using a 2 s threshold (arbitrarily chosen based on examining Figures 1 and 2), we observed that the overall mean keystroke time is around 195 ms, which falls between average (200 ms) and good (120 ms) typists. Some latency in keystroke times may be attributed to interruptions or other externalities not controlled in this environment.

Keystrokes

Figure 3 shows the distribution of the 1.512 million keystrokes by users meeting the criteria in Table 1 (95.6% meet the criteria). It shows that users type different amounts. Table 3 shows the relationship of dwell time between keystrokes and the words per minute. The mean keystroke time was 195.5 ms, equal to 296.7 characters per min. and 48.3 wpm. The median keystroke time was 191.2 ms. Figure 4 shows the distribution of keystroke times per user, with a minimum average time of 76.5 ms (128.5 wpm) and a maximum of 291.7 ms (33.7 wpm). For comparison, the base keystroke time of the KLM is 200 ms (49.2 wpm) for secretaries (professional) and 280 ms (35.1 wpm) for non-secretary (non-professional) typists (Card et al., 1983). Our findings suggest that users type faster than their cohorts in 1983.

Table 3: Relationship of keystroke times to words per minute (wpm)

Keystroke Time (ms)	Keys / Minute	Word Chars / Minute	Words / Minute
50	1200	984	197
75	800	656	131
100	600	492	98
150	400	328	65
200	300	246	49
250	240	197	39
300	200	164	33
350	171	141	28

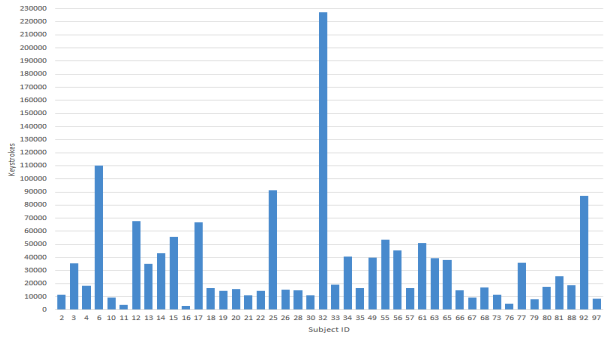


Figure 3: Distribution of keystrokes by user using 2-s cutoff. X-axis labels indicate participant IDs.

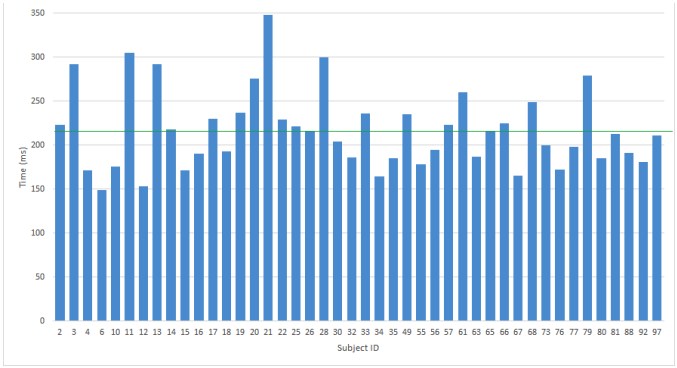


Figure 4: Distribution of mean keystroke times by user.

Homing

Users homed (moved their hand) from keyboard to mouse 51,989 times, and from mouse to keyboard 45,302 times (total: 97,291 times). Table 2 shows the criteria constituting homing. Combined, this rate is 53.6/hour or 0.89/minute. This action occurs once for every 15.4 keystrokes (using all keystrokes).

Figure 5a and 5b shows the distribution of homing times between the keyboard to mouse and vice-versa. The time to home took less time in either direction than the 0.40 s predicted by the KLM (Card et al., 1983). Homing had a mean value of 0.16 s for keyboard-to-mouse and 0.32 s for mouse-to-keyboard. The combined mean is 0.230 s. The

difference in the times between directions implies that the user needs additional time to return hands to the keyboard to continue typing; while when starting to mouse, the user may often be still typing while the hand is moving towards the mouse. These results suggest that the direction of homing (mouse to keyboard vs keyboard to mouse) are different actions, do not occur equally often, and could be separated into two distinct constants to give more accurate predictions.

We also observed a large number of occurrences—8051—of 0 ms homings (combined). These times may indicate co-joint typing and mousing or typing, or accidental trackpad movements.

Figure 5b and to a lesser extent Figure 5a, show a small peak in the distribution around 500-700 ms. The second peaks of the distribution were not expected; their existence might be due to two different user behaviors. We speculate that the 0-100 ms response time represents homing when one hand is on the keyboard and the other is simultaneously on the mouse, allowing near instantaneous homing, while the other response time represents a short delay between keystroke and homing.

Therefore, we see that movements between keyboard and mouse occur about once for every 15.4 keystrokes, and that this time is likely now done in two strategies: (a) while typing one moves the mouse with the other hand, and (b) stop typing to use the mouse with the last hand to type. The former is somewhat faster. We also saw that time to keyboard and time to mouse might be different actions. It might be useful to separate these two actions into separate actions to improve predictions where this is desirable.

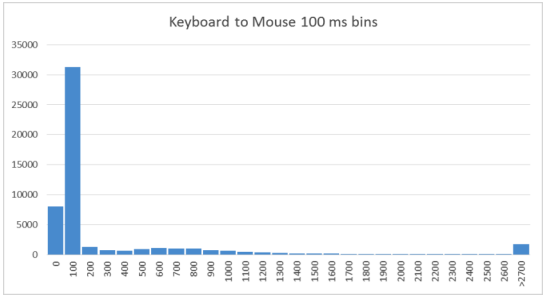


Figure 5a: Distribution of keyboard to mouse times; mean of the whole distrbution is 0.16 s.

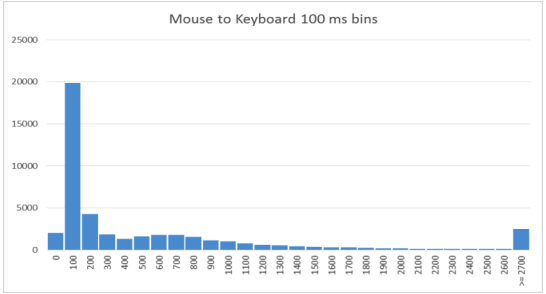


Figure 5b: Distribution of mouse to keyboard times; mean of the whole distrbution is 0.32 s.

We also saw that from the relative frequency of the two types of homing behaviors, that users tend to end a short set of unit tasks on the keyboard (there are more mouse to keyboard actions) than end a session with using the mouse (there are fewer keyboard to mouse actions).

There can be some ambiguity to what constitues homing, as the user may have their hand on the mouse causing a stream of mouse and keyboard data that still adheres to the criteria we proposed. Knowledge of the keys typed as well as the preferred mousing hand of the user could lessen this ambiguity. Furthermore, if the user is using a laptop with built in trackpad, the user could be mousing with their thumb and the remaining fingers not leaving the keys.

Conclusions

This study can only be a start to exploring naturalistic behavior with computers—there is a much greater range of environments and types of users than we could cover here. However, we were able to provide a general summary of naturalistic user behavior and to update several user constants used by many cognitive architectures. Naturalistic user studies of computer interaction can provide a new domain for large data analyses.

We found the definitions of these measures more important than we initially thought. For example, the definition of typing or homing depends on more than just dwell time. There were many additional hidden aspects. For instance, how do we detect when a user was away from the keyboard? When did the user switch from the keyboard to the mouse? Was the user moving the mouse and typing at the same time or was there desk instability that lead to the mouse appearing to have moved whilst the user was typing?

Updating the KLM Constants

We saw in this naturalistic study that the typing rate is sensitive to typing threshold. We can use this approach in later analyses. This result also reminds us that typing speed is not a fixed rate but a distribution, and that the tails of this distribution might be important for some analyses.

The amount of typing over a workday is not very large, (average of about 7,700 keystrokes). Subsequently, the typing rate over a workday is not fast (about 2 wpm). While the amount of typing varied by user by over a factor of 20, the typing rate varied by only a factor of 2.

These results suggest that users are typing faster than Card et al.'s (1980) KLM expert times, and that current naturalistic typing is not much slower than previous timed typing.

We suggest that architectures that use these constants (e.g., ACT-R, EPIC) (a) update their typing rate, (b) allow for additional concurrent motor commands, (c) split homing actions into two distinct actions, and (d) use these shorter times for homing tasks.

Limitations

While we include a range of faculty, staff, and students, our participants were all from the same university environment.

A wider selection of users is likely to provide a wider range of behavior. Users from different work domains, a greater number of users using computers more at home or in other environments, and older and younger users might lead to a different constants and different results. It may be useful to rerun this study with those more varied types of users.

While we analyzed a large amount of data, observing additional users could provide more support for these conclusions.

We were limited by the anonymization of the data: the keystroke logger anonymized many of the details of the data making it difficult to determine what, task the users were performing. While anonymization was necessary for the protection of participants' privacy, with non-character key data such as return, arrow keys, whitespace, home key, etc., we would have determined more conclusively the tasks performed. For example, was the user typing a document, or programming; playing a game that required the use of arrow keys, or browsing a webpage and pressing page down to browse? The difference between tasks could assist in explaining the differences in rate and quantity between users. For example, space and enter key laden tasks can be more time consuming from a typing perspective, as the space and enter keys require more time to type (300 ms and 550 ms on average respectively) (Kinkead, 1975; Ostry, 1983). Alternatively, users who are browsing may home more frequently.

Furthermore, anonymization made it impossible to determine error rate, another interesting result that could help design systems. Logging editing keys may have provided context that would indicate when errors occurred, through a series of delete or backspace keys following a number of character and space keystrokes.

Further Studies

Our analyses performed led to further questions concerning how users interact with computers. Further research that would yield valuable insight includes studies of the use of shortcut key chords and non-anonymized or less anonymized keystrokes. Having associated details with semi-non-anonymized keystroke data could eliminate potential artifacts of simple document browsing, web browsing, or game playing and provide novel insights into how people multitask.

As wireless devices become increasingly prolific, research centered on touchscreen input and touchscreen typing would also be fascinating. The difference in these devices between manufacturers and the lack of a quality keystroke logger would constitute a limiting factor for completing this research, however.

Acknowledgments

Portions of this work were funded by ONR (N00014-15-1-2275 & N00014-11-1-0275). Chris Dancy, P. Greg Plumb, and Jon Morgan helped gather this data. We thank our colleagues who gave us their time and keystroke logs, and Ysabelle Coutu and Dan Guzek for comments on this paper.

References

Card, S. K., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Erlbaum.

Card, S. K., & Moran, T. P. (1986). User Technology: From pointing to pondering. Proceedings of HPW '86, Proceedings of the ACM Conference on The history of personal workstations, 183-198.

Card, S. K., Moran, T. P., & Newell, A. (1980). The Keystroke-Level Model for user performance time with interactive systems. *Communications of the ACM, 23*(7), 396-410.

Kieras, D. E. (1988). Towards a practical GOMS model methodology for user interface design. In M. Helander (Ed.), *Handbook of Human-Computer Interaction* (pp. 135–158). Amsterdam: North–Holland Elsevier.

Kinkead, R. (1975). Typing speed, keying rates, and optimal keyboard layouts. In Proceedings of the Human Factors Society 19th Annual Meeting 159-161. Human Factors Society: Santa Monica, CA.

Kukreja, U., Stevenson, W. E., & Ritter, F. E. (2006). RUI—Recording User Input from interfaces under Windows and Mac OS X. *Behavior Research Methods, 38*(4), 656–659.

MacKenzie, I. S. (2002). KSPC (Keystrokes per Character) as a characteristic of text entry techniques. *Proceedings of the Fourth International Symposium on Human Computer Interaction with Mobile Devices*, 195-210.

MacKenzie, I. S., Sellen, A., & Buxton, W. (1991). A Comparison of input devices in elemental pointing and dragging tasks. *Applied Ergonomics, 23*(6), 161-166.

Morgan, J. H., Cheng, C.-Y., Pike, C., & Ritter, F. E. (2013). A design, tests, and considerations for improving keystroke and mouse loggers. *Interacting with Computers, 25*(3), 242-258.

Ostry, D. J. (1983). Determinants of interkey times in typing. *Cognitive aspects of Skilled typewriting*, 225-246.

Salthouse, T. A. (1984). Effects of age and skill in typing. *Journal of Experimental Psychology: General, 113*(3), 345-371.

Salvucci, D. D., & Taatgen, N. A. (2011). *The multitasking mind*. New York: Oxford.

Spink, A., Ozmutlu, H. C., & Ozmutlu, S. (2002). Multitasking information seeking and searching processes. *Journal of the American Society for Information Sciences and Technology, 53*(8), 639-652.

Whisenand, T. G., & Emurian, H. H. (1999). Analysis of cursor movements with a mouse. *Computers in Human Behavior, 15*(1), 85-103.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Exploring the Effects of Different Text Stimuli on Typing Behavior

Ignacio X. Domínguez (ignacioxd@ncsu.edu), **Jayant Dhawan (jdhawan2@ncsu.edu)**,
Robert St. Amant (stamant@csc.ncsu.edu), **David L. Roberts (robertsd@csc.ncsu.edu)**
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206 USA

Abstract

In this work we explore how different cognitive processes affected typing patterns through a computer game we call *The Typing Game*. By manipulating the players' familiarity with the words in our game through their similarity to dictionary words, and by allowing some players to replay rounds, we found that typing speed improves with familiarity with words, and also with practice, but that these are independent of the number of mistakes that are made when typing. We also found that users who had the opportunity to replay rounds exhibited different typing patterns even before replaying the rounds.

Keywords: Typing; Cognitive Model; Keyboard; Input Device; Cognition; Speed-Accuracy Trade-off; Speed; Accuracy; Inter-keystroke Interval; Typing Mistakes; Human-Computer Interaction; Human Information Processing; Human Behavior.

Introduction

In this paper we present early exploratory work toward creating cognitive models of interaction from patterns in typing that can be applied to, among other things, identifying the cognitive processes at play when users are typing on keyboards. In particular, we focus on the use of a keyboard as a window into interaction patterns that are reflective of the user's cognitive state. By mining for patterns on the usage of input devices we aim to unobtrusively obtain a snapshot of users' perception and decision-making processes in real-time.

To explore typing patterns and their relationship to cognition we created a computer game that involved typing words of different lengths with varying *word shapes* (Bouwhuis & Bouma, 1979). Some participants were able to replay rounds. We recorded typing speed and accuracy expecting an improvement as the rounds were replayed, as well as better speed and accuracy while typing words more similar to dictionary words. By changing the nature of the words being typed, we were able to alter the cognitive process required to type them, allowing us to measure how the differences in cognition are reflected in typing patterns. Our overarching goal beyond this paper is to be able to create models of interaction that would allow real-time detection of the user's cognitive state across a wide variety of tasks and interfaces.

Related and Prior Work

Computational Cognitive Modeling

When typing or pointing at targets in a graphical user interface, users exhibit distinctive patterns in the timing of their keystrokes (Monrose & Rubin, 1997) and the movement of the mouse (Ahmed & Traore, 2007). At a more abstract level, human decision-making has also been studied. A range of "microstrategies" (Gray & Boehm-Davis, 2000) applied to low-level HID usage have been identified. Microstrategies are characteristic choices that users make, without extensive deliberation, between different actions to achieve their goals.

A representative example of research on microstrategies is due to Gray and Fu (2004), where participants were given a task to perform in a user interface that contained an information box, with a variable cost of accessing that information in different conditions: the information could be permanently visible or it could require a mouse click (with a temporary lockout) to see. They found patterns in completion times, error rates, and decisions made by participants across the conditions, which could be explained in terms of trade-offs between perceptual/motor and memory retrieval effort. Participants' behaviors depended in subtle ways on cognitive biases (*e.g.*, a preference for "knowledge in the head" rather than perfect knowledge in the world).

Models of Typing

Transcription typing has been well-studied, with some work looking at how typing speed varies with unfamiliar material. Salthouse (1986) observes 12 "basic phenomena" about typing, one of which describes the reduction in the typing speed when the typist is presented with random sequences of letters.

John (1996) introduced the TYPIST model that "can be used to make quantitative predictions of performance on typing tasks". This model is based on the Model Human Processor (MHP) (Card, Moran, & Newell, 1986). TYPIST applies the MHP to human typing tasks for skilled transcription typists in order to quantitatively predict the performance of the typists. It processes text at the level of *chunks*, which could be words, syllables, or letters. TYPIST is applied to several common typing tasks, and its predictions of the performance of the typists come to within 20% of empirical measurements.

While previous work has focused on skilled or "expert" typists, little work has explored typing patterns of average users. While Feit, Weir, and Oulasvirta (2016) recently explored the mechanics and strategies of everyday typists, the cognitive processes involved in typing different types of content remains largely unexplored.

Method

Because of the exploratory nature of this work, we focused primarily on establishing internal validity. Our main goal was to get insight into how the cognitive processes associated with typing change in relation to changes in what is being typed and previous exposure to the material. Our approach focuses on the use of computer games that elicit examples of different

typing behaviors. In particular, we designed and implemented *The TypingGame*—a casual game that we present below.

Using a computer game provides some advantages at this early stage of our research. First, because game mechanics often result in changes to the details of tasks, users tend to be more accepting of changes to an interface or expectations on their performance. Second, and perhaps more importantly, computer games provide motivational context. In order to get reasonable data, users had to have an incentive to perform the task well. The "gameification" of the task enabled us to study users under experimental conditions with relatively higher engagement when compared to a stand-alone typing task.

Target Population and Sampling

We targeted computer users of at least 18 years of age, and recruited using a combination of convenience and snowball sampling. We advertised our study primarily to the Computer Science student body at the authors' institution, but also posted fliers on nearby bulletin boards. Participants were offered a base compensation of $5.00 and a maximum of an additional $2.00 for each game round they completed, for a total maximum of up to $25.00 based on their gameplay performance. Interested individuals were asked to sign up online for an available time slot and location.

Our sample consisted of 43 participants, of which 14 were female and 29 were male. Before the game, we asked the participants to rate their typing skills by choosing one of these options: *Beginner*, *Intermediate*, *Advanced*, or *Expert*. Of the females, 8 reported their skills as intermediate, and 6 as advanced. In the case of the male participants, 2 reported their skills as beginner, 14 as intermediate, and 13 as advanced. The average age of the female participants was 23.57 ($SD = 2.53$) years, and for the males, it was 23.90 ($SD = 2.13$) years.

The Typing Game

Our implementation of *The Typing Game* was written in Adobe Flash CS5.5 and was designed to run on a Web browser. The goal of the game is to type words that are shown on a 4×4 game board grid as fast as possible. Sets of between 1 and 4 words, initially shown on the first row of the grid, one per column, drop down one row at periodic intervals until they are correctly typed or fall off the board. Words that are correctly typed immediately disappear from the board. If a mistake is made while typing a word, the word resets and must be typed again starting with the first letter.

The Game Experience Upon launch, our game randomly assigned the player to one of three experimental conditions. To ensure that the game screen had input focus and that the keyboard input was received by our game interface, the first screen prompted the player to press the SHIFT-N key combination to begin and presented a small demographics survey. The following screen presented a small survey that asked about the player's background and typing habits. Next, the game asked the participant to type the sentence "*the quick brown fox jumps over the lazy dog*". This sentence was used to ensure that the keyboard was working properly. Once this sentence was correctly typed, the player was prompted to press the SHIFT-N key combination to proceed to an in-game tutorial. The player was asked to press the space key to begin the tutorial, which started by explaining the game mechanics in an interactive manner prompting the player to type the word "go" in order to move to the next screen. This illustrated how words were removed from the game board once they were correctly typed. The next screen in the tutorial illustrated how sets of words would drop from the row on every time interval, and how drops affected scoring.

Table 1: Description of the rounds in our Typing Game.

Round	Round	Word Length	Word Type
Practice	Practice	Short	Dictionary word
1	DictM	Medium	
2	ShapeS	Short	Transposed letters preserving word shape
3	ShapeM	Medium	
4	ShapeL	Long	
5	NoShapeS	Short	Transposed letters breaking word shape
6	NoShapeM	Medium	
7	NoShapeL	Long	
8	RandS	Short	Random letters
9	RandM	Medium	
10	RandL	Long	

Next, the game introduced participants to a practice round that accurately simulated the mechanics that the player would experience in the game rounds. In order to advance to the game rounds, the player was required to earn at least $1.70 during the practice, and was required to replay the round until she did. The money earned during the practice round did not count towards the participant's final compensation.

To ensure that players were ready, each round had a staging screen that prompted the participant to press the space key to begin. After each round, a summary screen presented the round number, the amount earned, and a prompt to press the SHIFT-N key combination in order to proceed (some conditions also displayed a prompt to press the SHIFT-R key combination to replay the round, as described below). In addition to the practice round, participants completed a total of 10 game rounds, which did not require a minimum score.

Rounds A single game round contains multiple word sets that initially appear on the first row, but on different columns, of the game board grid. Our game consisted of 10 rounds varying the type of words and their length (see Table 1).

We designed our rounds with four types of words, all in lowercase: 1) dictionary words (*e.g.*, "quit") , 2) dictionary words with one or more transposed letters, preserving the general shape of the word (*e.g.*, "tiem" for time), 3) dictionary words with one or more transposed letters, breaking the general shape of the word (*e.g.*, "gluf" for gulf), and 4) words composed of random letters, filtering out common bi-grams and tri-grams to avoid confounding our variables. The idea behind the differences in word choices was to explore how the similarity of the word being typed to a real word affected the typing patterns. For the same reason, our rounds had different word lengths (short, medium, and large, as shown in Table 1, with 3-4, 4-5, and 5-6 characters, respectively).

Scoring Every round begins at the highest score ($2.00) and decreases by $0.05 (until it reaches $0) for every time a set of

untyped words drops down one row. For a player to earn the maximum score, she has to type every word correctly while they are still on the first row. The amount to be earned for a round is displayed at the bottom right of the game screen and is updated as the words drop.

Depending on the experimental condition, some players had the option to replay rounds by pressing the SHIFT-R key combination during a round's summary screen. The score earned for a round would be the one obtained on the last replay of that round, regardless of whether it was lower or higher than the score obtained in previous attempts.

Visual Design The 4×4 game board grid has a black background, where each cell is 200px wide and 100px tall. A cell with an untyped word will have a gray background. Every word uses the Consolas font in 18 point. The color of the font is initially black, but as a word is typed, the color of correctly-typed letters changes to a dark gray to show progress.

Experimental Procedure

The researchers asked participants to meet them at a designated room during a time slot previously agreed upon. After providing informed consent, participants were given the opportunity to ask questions before moving on to the data collection phase. At this time, the researchers would instruct participants to sit in front of a computer that was previously set up to run our game using the Google Chrome browser in full-screen mode. This computer was instrumented with a USB Microsoft Wired Keyboard 600 configured with US American visual and functional keyboard layouts. The researchers asked participants to notify them once they reached the final screen of the game and stepped out of the room, leaving the participants alone with no distractions.

Once they completed the game, participants would notify the researchers who would then record the participant's earnings and a unique game-generated code from the last game screen onto a paper form that participants would later use to collect their compensation. The purpose of this last step was to avoid associating participants with the data that was collected from their participation.

Experimental Conditions

Participants were randomly assigned to one of three experimental treatments.

- **Replay not allowed:** Participants were not allowed to voluntarily replay any rounds. The practice round could only be replayed until the minimum score of $1.70 was obtained. The summary screen of every round only allowed participants in this treatment to advance to the next round.
- **Replay encouraged:** Participants were allowed to voluntarily replay any round an unlimited amount of times, including the practice round after the minimum score of $1.70 was obtained. The summary screen of every round showed both the key combination to press in order to advance to the next round and the key combination to press in order to replay the round.
- **Replay allowed:** Participants were allowed to voluntarily replay any round an unlimited amount of times, including the practice round after the minimum score of $1.70 was obtained. The summary screen of every round showed both the key combination to press in order to advance to the next round and the key combination to press in order to replay the round, but the latter was displayed as if it were inactive (grayed out) despite being functionally equivalent to the **replay encouraged** treatment.

We found that participants in the *Replay allowed* treatment never attempted to replay a game round and therefore behaved in the same way as participants in the *Replay not allowed* treatment. We believe that displaying the replay prompt as inactive was enough to make participants believe that they did not have the ability to use that feature. For the purpose of our analyses, we will treat all participants in these two treatments as a single group. We will refer to these groups as the **replay** (16 participants) and **no replay** (27 participants) conditions based on whether they voluntarily replayed rounds or not, respectively.

Logs and Analytics

We had three independent variables in our experiment: 1) the *Round* of our game being played, which modified the length and type of words that players had to type, 2) the *Condition*, which dictated player's ability to replay rounds, and 3) the *Attempt*, which indicates how many times the round is being replayed. In the case of the no replay condition, the value of the *Attempt* of a game round is always 1.

Our implementation of *The Typing Game* captured the "key down" and "key up" keyboard events, causing each keystroke to be recorded as two events. In addition to the key that generated the event, our game also collected a timestamp, with millisecond precision, of when each event occurred. Each key event was also associated to the round or screen active when it occurred, to any word on the board to which it may have corresponded, whether the keystroke was correct or not, and whether it completed a word on the board. These low-level data allow us to calculate higher level metrics. In particular, in this paper we define the following analytics:

- **Inter-keystroke interval (IKI):** the number of milliseconds elapsed between the "key down" events of each contiguous pair of keystrokes in a correctly-typed word. For the purposes of this metric, we excluded events from words that were typed with mistakes.
- **Number of mistakes:** the count of keystrokes during a round that did not clear the game board, or that did not result in the board being one character closer to being cleared. For the purposes of this metric, we did not count whitespace characters as mistakes.

The IKI is a common metric for typing speed (Salthouse, 1986), while the number of mistakes is a natural metric for typing accuracy. We will refer to *typing speed* as the inverse of the IKI, where a smaller IKI represents an increase in speed (and vice versa), and to *typing accuracy* as the inverse of the

number of mistakes made, where fewer mistakes indicate a higher accuracy (and vice versa).

Formally, our hypotheses are:

H1: Practice increases speed – The average IKI in a round will be smaller when replaying.

H2: Practice increases accuracy – The average number of mistakes in a round will be smaller when replaying.

H3: Familiar words are typed faster – The average IKI of a word will be smaller the closer the word is to a dictionary word.

H4: Familiar words are typed more accurately – The average number of mistakes made when typing a word will be smaller the closer the word is to a dictionary word.

Analysis and Results

To ground the internal validity of our study with respect to both speed and accuracy, we compared the first attempt of the practice round between both the replay and no replay conditions. Because the game experience for both conditions is identical at this point in the game, we expected no substantial difference between the two. To evaluate the significance of the difference in the average IKI between the replay ($M = 164.32, SD = 92.45$) and no replay ($M = 163.62, SD = 121.77$) conditions on the first attempt of the practice round, we conducted a Welch's independent-samples t-test, which revealed no significant difference in speed ($t(1170.2) = -0.12995, p = 0.8966$). To evaluate the significance of the difference in the average number of mistakes between the replay ($M = 5.44, SD = 5.51$) and no replay ($M = 5.78, SD = 5.89$) conditions on the first attempt of the practice round, we conducted a Welch's independent-samples t-test, which revealed no significant difference in accuracy ($t(33.331) = 0.19074, p = 0.8499$).

Having established the first attempt of the practice round as a valid baseline across our experimental conditions, we used the individual player's averages of IKI and number of mistakes on this attempt of this round to normalize their game rounds' IKI and number of mistakes, respectively, by dividing the measured value by the average. We use these normalized values for the rest of our analyses. Descriptive statistics for IKI and number of mistakes made in each round by attempt are shown in Table 2 and Table 3, respectively.

Improvement with Practice

This part of the analysis focuses on the replay condition as it was the only one that allowed replaying rounds. Even though participants in the replay condition were allowed to replay as many times as they wanted, the most participants replayed a single round was 8 times. However, because at most 3 participants replayed a single round more than 4 times, we decided to focus on the first 4 attempts in our analysis.

Our hypothesis **H1** expects there to be an improvement in speed as rounds are replayed. We conducted a factorial ANOVA to examine the effects of Attempt and Round on the IKI. The results yielded a main effect for the attempt ($F(1, 15471) = 102.4765, p < 0.001$), indicating that the typing speed of participants significantly increased (*i.e.,* the IKI decreased) the more rounds were replayed. The main effect of the round was also significant ($F(9, 15471) = 102.4765, p < 0.001$). The interaction effect was non-significant ($F(9, 15471) = 0.8436, p > 0.1$). This results is consistent with our hypothesis **H1**.

Table 2: Normalized mean and standard deviation of the inter-keystroke interval of participants on the "replay" condition on each of the first four attempts of every round.

	Attempt 1		Attempt 2		Attempt 3		Attempt 4	
	M	**SD**	**M**	**SD**	**M**	**SD**	**M**	**SD**
DictM	1.03	0.58	1.02	0.61	0.96	0.63	0.96	0.41
ShapeS	1.22	0.69	1.18	0.60	1.12	0.77	1.15	0.48
ShapeM	1.29	0.90	1.24	0.95	1.13	0.70	1.05	0.55
ShapeL	1.52	1.07	1.40	0.92	1.36	0.86	1.25	0.75
NoShapeS	1.21	0.65	1.15	0.55	1.06	0.50	1.05	0.60
NoShapeM	1.37	0.94	1.26	0.72	1.19	0.72	1.11	0.68
NoShapeL	1.46	0.99	1.42	0.82	1.21	0.64	1.22	0.75
RandS	1.41	0.89	1.25	0.72	1.18	0.55	1.24	0.80
RandM	1.68	1.36	1.58	1.03	1.37	0.85	1.54	1.50
RandL	1.95	1.49	1.91	1.45	1.77	1.14	1.56	1.03

Table 3: Normalized mean and standard deviation of the number of mistakes made by participants on the "replay" condition on each of the first four attempts of every round.

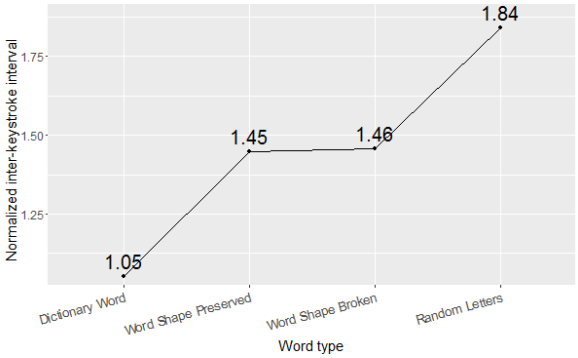
	Attempt 1		Attempt 2		Attempt 3		Attempt 4	
	M	**SD**	**M**	**SD**	**M**	**SD**	**M**	**SD**
DictM	2.85	2.74	2.98	2.86	2.98	3.15	2.2	N/A
ShapeS	3.17	3.85	2.40	1.69	2.24	1.043	2.2	0.28
ShapeM	2.13	2.07	5.07	4.20	3.39	2.94	5	N/A
ShapeL	5.68	7.47	5.09	3.35	3.08	1.88	4.38	4.14
NoShapeS	2.46	2.18	2.69	1.54	2.65	2.07	4.65	5.93
NoShapeM	3.767	3.48	3.30	1.97	3.06	1.65	5.05	0.64
NoShapeL	6.19	6.63	4.10	2.34	5.17	3.10	6.44	5.24
RandS	1.91	2.06	1.99	1.90	1.87	1.46	2.18	1.78
RandM	4.94	3.94	4.03	2.13	5.48	1.90	4.37	4.20
RandL	4.72	4.89	2.71	1.81	4.03	3.52	6	2.83

Our hypothesis **H2** expects there to be an improvement in accuracy as rounds are replayed. As before, we conducted a factorial ANOVA to examine the effects of Attempt and Round on the number of mistakes made. The results yielded a main effect for the round ($F(9, 284) = 3.2348, p < 0.001$), indicating that the typing accuracy of participants is significantly dependent on the round that was being played. The main effect of the attempt was not significant ($F(1, 284) = 0.0693, p > 0.1$). The interaction effect was also non-significant ($F(9, 284) = 0.4621, p > 0.1$). This results contradicts our hypothesis **H2**.

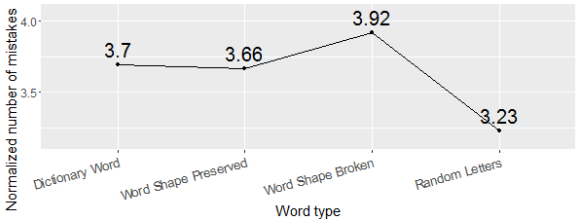
Familiarity with Words

For this analysis we look at how the different types of words in our game rounds affected speed and accuracy. In particular, we expected words that are more similar to real words to

be typed faster (**H3**) and more accurately (**H4**). In decreasing order of similarity to real words we have dictionary words (*DictM*), dictionary words with transposed letters preserving the shape of the word (*ShapeS*, *ShapeM*, and *ShapeL*), dictionary words with transposed letters breaking the shape of the word (*NoShapeS*, *NoShapeM*, and *NoShapeL*), and random letters (*RandS*, *RandeM*, and *RandL*).



(a) Inter-keystroke interval



(b) Number of mistakes

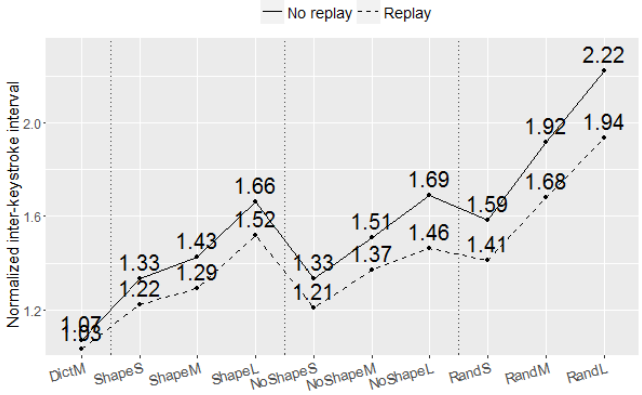
Figure 1: Comparison of the average normalized inter-keystroke interval and normalized number of mistakes by word type on the first attempt of every round.

The average IKI increases as the words participants typed resembled less real words (see Figure 1(a)), as predicted by **H3**. To evaluate significance of this difference we conducted a factorial ANOVA that explored the effects of word length, word type, and condition on IKI. The results yielded statistically significant interactions between the word type and word length ($F(4,28374) = 22.9631, p < 0.001$), between word length and condition ($F(2,28374) = 9.2675, p < 0.001$), and between word type and condition ($F(3,28374) = 10.4835, p < 0.001$). The interaction between word length, word type, and condition was not significant ($F(4,28374) = 0.1962, p > 0.1$). Simple main effects analysis showed significant differences in speed dependent on word length ($p < 0.001$), word type ($p < 0.001$), and condition ($p < 0.001$). This result is consistent with hypothesis **H3**.

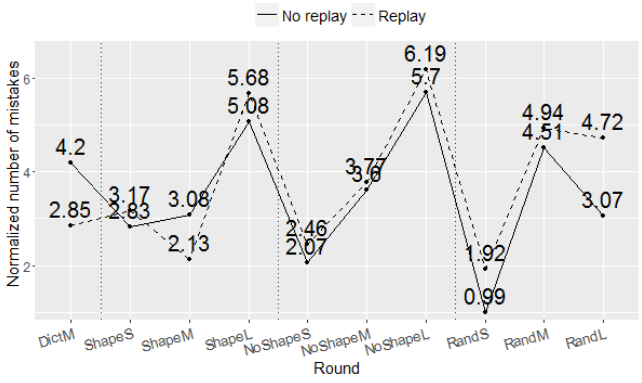
Our hypothesis **H4** expects participants to be more accurate on words that are closer to dictionary words. Figure 1(b) shows the number of mistakes made by our participants according to the type of word being typed. To evaluate these differences we conducted a factorial ANOVA that explored the effects of word length, word type, and condition on the number of mistakes made. The results yielded a statistically significant interaction between the word type and word length ($F(4,554) = 3.0836, p = 0.01578$). All the other interactions were not significant. Simple main effects analysis showed a significant difference in accuracy dependent on word length ($F(2,554) = 16.8025, p < 0.001$). We found no statistically significant difference in accuracy dependent on the type of the word. The lack of significance of the effect of the word type contradicts our hypothesis **H4**.

Additional Analyses

To obtain more insight we ran additional tests to compare the replay and no replay conditions on both speed an accuracy metrics, both on the first attempt of every round, and with up to 4 replays (for the replay condition; the no replay condition only had one attempt per round).



(a) Inter-keystroke interval



(b) Number of mistakes

Figure 2: Comparison of the average normalized inter-keystroke interval and normalized number of mistakes by condition on the first attempt of every round. The vertical lines separate rounds by word type.

When comparing the first attempt of every round between conditions we found that the mean IKIs of the replay condition were consistently smaller than those of the no replay condition (see Figure 2(a)). Using a Welch's independent-samples t-test, we found a significant difference in speed on the first attempt of every round between conditions ($t(18384) = 10.236, p < 0.001$).

In contrast, as shown on Figure 2(b), we don't see a clear distinction when comparing the number of mistakes made

on the first attempt of every round between conditions. To determine significance difference we conducted a Welch's independent-samples t-test, which revealed no significant difference in accuracy on the first attempt of every round between conditions ($t(350.59) = -0.59744, p = 0.5506$).

Discussion

The above analysis confirms that typing speed improves with practice and when the words are more familiar. Surprisingly, we find that this improvement in speed is not accompanied by an improvement in typing accuracy neither with practice nor with familiarity with the words being typed. The number of mistakes made cannot be used to explain the reduction in IKI.

We saw that on the first attempt of the practice round, where the game experience is identical between conditions, all of our participants behave similarly. However, as the game progresses, participants in the replay condition significantly increase their typing speed without any improvement in the number of mistakes they make, indicating that the speed improvement is not attributable to an increase in accuracy. Because the only difference between conditions is the ability to replay rounds, a plausible explanation for this behavior lies in the fact that the cost (in terms of mathematical utility) of making mistakes is smaller than the reward of earning a higher compensation by typing faster, because the opportunity to replay the round is always there. This behavior is consistent with research on task accomplishment strategies, where there exists a trade-off between speed and accuracy (Barik, Chakraborty, Harrison, Roberts, & Amant, 2013; Gerjets, Scheiter, & Tack, 2000; Heitz, 2014). We find support for this explanation in our data when we compare the typing speed on the first attempt of every level between the replay and no replay conditions (see Figure 2(a)). On these first attempts, players in both conditions have had the same exposure to the words on each round, ruling out familiarity as an explanation for the significant difference in speed between conditions. We see that participants in the replay conditions are consistently and significantly faster than participants in the no replay condition after being exposed to the possibility of replaying, whereas this difference is non-existent on the first attempt of the practice round where they have not been exposed to this game mechanic.

Our results show that speed has a more direct relationship to the nature of what is being typed than the number of mistakes that are made while typing. This suggests that by inspecting typing speed a system can be more effective at detecting anomalies (and possibly identifying the cause of the anomaly) than looking at the number of incorrect attempts alone. Similarly, our results indicate that typing speed can be used to identify the familiarity to the text being typed, which can be used to compare to a known baseline.

Conclusions

In this work we explored how different cognitive processes affected typing patterns by manipulating the similarity of words to dictionary words, and by allowing participants to replay rounds of *The Typing Game*. We found that the typing speed improves with familiarity with words and with practice, but that these are independent of the number of mistakes that are made when typing. We also found that users exhibit different typing patterns when they are made aware of a penalty for mistakes than when they don't expect consequences for mistakes. Our results allow us to better understand the cognitive processes involved in typing.

There are several limitations to consider when interpreting our results. As mentioned earlier, we focused on establishing internal validity of our study, giving our first steps toward building cognitive models of input device interaction patterns. Firstly, because our sample was comprised mostly of Computer Science students, the typing proficiency of our participants is probably well above average, which is a threat to the external validity of our findings. Secondly, our game did not attempt to establish ecological validity, but was instead designed to elicit specific behaviors that manipulated the cognitive processes required to complete the game rounds. Thirdly, the nature of the words included in our game was also intentionally limited, and did not include numbers, uppercase letters, nor special characters. Despite these limitations, the empirical data we collected will allow us to generate cognitive models from interaction patterns of real users that can then be validated with a more representative sample and on multiple domains, pointing to avenues for future work.

Future Work

The data we collected from *The Typing Game* is incredibly rich, and this work presents preliminary results that we will use as stepping stones toward creating the cognitive models we discussed. We have already started working on a playback and visualization tool that will enable us to not only inspect and tag our data in more detail, but also to tweak and validate our assumptions as our models are created. We would like to explore how typing patterns differ with a more diverse character set, such as including capitalization, special characters and punctuation, and texts of different lengths (*e.g.,* a paragraph instead of a single word).

With a better understanding of typing phenomena and their relationship to cognition, we expect to validate our models on multiple domains. In particular, we aim to validate our models in domains that more closely resemble real-world tasks.

We also plan to investigate cognitive models of different input device usage. We expect that certain domains would benefit from models of multiple input devices simultaneously in order to improve the accuracy of their predictions, but also in order to provide a richer characterization of usage patterns.

Acknowledgments

This work was funded in part by the National Security Agency (grant number H98230-14-C-0139) and in part by the National Science Foundation (grant number IIS-1451172). The authors thank Brent Harrison and Arpan Chakraborty for their contributions to the design of *The Typing Game*.

References

Ahmed, A. A. E., & Traore, I. (2007). A new biometric technology based on mouse dynamics. *IEEE Transactions on Dependable and Secure Computing*, *4*(3), 165–179.

Barik, T., Chakraborty, A., Harrison, B., Roberts, D. L., & Amant, R. S. (2013). Speed/accuracy tradeoff in act-r models of the concentration game. In *Proceedings of the 2013 international conference on cognitive modeling* (pp. 281–286).

Bouwhuis, D., & Bouma, H. (1979). Visual word recognition of three-letter words as derived from the recognition of the constituent letters. *Perception & Psychophysics*, *25*(1), 12–22. doi: 10.3758/BF03206104

Card, S. K., Moran, T. P., & Newell, A. (1986). The model human processor: An engineering model of human performance. *Handbook of Perception and Human Performance*, *2*, 1–35.

Feit, A. M., Weir, D., & Oulasvirta, A. (2016). How we type: Movement strategies and performance in everyday typing. In *Proceedings of the 2016 chi conference on human factors in computing systems* (pp. 4262–4273). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2858036.2858233` doi: 10.1145/2858036.2858233

Gerjets, P., Scheiter, K., & Tack, W. H. (2000). Resource-adaptive selection of strategies in learning from worked-out examples. In *Proceedings of the 22nd annual conference of the cognitive science society* (pp. 166–171). Erlbaum.

Gray, W. D., & Boehm-Davis, D. A. (2000). Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied*, *6*(4), 322–335.

Gray, W. D., & Fu, W. T. (2004). Soft constraints in interactive behavior: The case of ignoring perfect knowledge in-the-world for imperfect knowledge in-the-head. *Cognitive Science*, *28*(3), 359–382.

Heitz, R. P. (2014). The speed-accuracy tradeoff: history, physiology, methodology, and behavior. *Frontiers in neuroscience*, *8*(150). doi: 10.3389/fnins.2014.00150

John, B. E. (1996). Typist: A theory of performance in skilled typing. *Hum.-Comput. Interact.*, *11*(4), 321–355.

Monrose, F., & Rubin, A. (1997). Authentication via keystroke dynamics. In *Proceedings of the 4th acm conference on computer and comms security* (pp. 48–56).

Salthouse, T. A. (1986). Perceptual, cognitive, and motoric aspects of transcription typing. *Psychological Bulletin*, *99*(3), 303–319.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Efficient Computation of Spreading Activation Using Lazy Evaluation

Steven J. Jones (scijones@umich.edu)
Arthur R. Wandzel (awandzel@umich.edu)
John E. Laird (laird@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121

Abstract

Spreading activation is an important component of many computational models of declarative long-term memory retrieval but it can be computationally expensive. The computational overhead has led to severe restrictions on its use, especially in real-time cognitive models. In this paper we describe a series of successively more efficient algorithms for spreading activation. The final model uses lazy evaluation to avoid much of the computation normally associated with spreading activation. We evaluate its efficiency on a commonly-used word-sense disambiguation task where it is significantly faster than a naive model, achieving an average time of 0.43ms per query for a spread to 300 nodes.

Keywords: cognitive architecture; context-sensitive retrieval; Soar; semantic memory; spreading activation.

Introduction

As cognitive modeling moves to more complex and real-world tasks, there is a challenge of maintaining efficient, scalable, context-sensitive access to long-term knowledge. In prior research, our group developed efficient and scalable algorithms for context-free cue-based retrievals (Derbinsky, Laird, & Smith, 2010). In this paper, we extend that work to context-sensitive retrievals by developing efficient and scalable algorithms for spreading activation (Anderson, 1983b).

In cognitive architectures, such as Soar (Laird, 2012) and ACT-R (Anderson, 1983a), working memory defines the *context*—the agent's task-relevant knowledge. Spreading activation supports context-sensitive retrieval by biasing the retrieval of elements from long-term declarative memory to those that have direct and possibly indirect long-term associations to structures in working memory. Previous work has focused on mitigating the cost of spreading activation through using high-performance computers, external data-base technology, and parallelism (Douglass & Myers, 2010; Chen, Petrovic, & Clark, 2014; Edmonds, Atahary, Taha, & Douglass, 2015). The expense of spreading activation has led cognitive modelers to severely limit the depth of spread or to avoid it completely. Efficient spreading could dramatically increase its use in cognitive models and decrease the time it takes to run simulations. It could also enable spreads to greater depth, so as to extract knowledge that is latent within the structure of an agent's long-term memory. Finally, efficient spreading would support its use in real-time cognitive models and AI agents.

As in our prior work, our investigations are within Soar. Soar provides an efficient platform, both in terms of overall performance, but more specifically in terms of an efficient implementation of long-term declarative memory. For cognitive modeling, Soar's decision cycle corresponds to the production firing cycle of ACT-R, which maps to approximately 50ms of human behavior. However, on a standard workstation, Soar's decision procedure runs at less than 0.3ms, even with large numbers of rules and declarative memory elements. The essence of this paper is adding spreading activation to Soar with a simple naive algorithm, then reconceptualizing that algorithm through a series of optimizations. Those optimizations take advantage of important regularities in the dynamics of Soar's long-term semantic memory. The ratio of changes to the context (working memory) to the total number of elements in the context is small. The ratio of long-term memory changes to the total number of elements in the long-term memory is even smaller. Many queries of long-term memory are unambiguous and even those with ambiguity are often constrained to only a few possibilities. We evaluate these optimizations on a word-sense disambiguation task that has proven usual for evaluating efficiency of long-term memory retrieval in the past (Derbinsky & Laird, 2011).

Background

In the Soar cognitive architecture, working memory maintains an agent's current knowledge of its task and environment, including active goals, results of perception, inferences, and retrievals from long-term memory. Behavior is conditional on the contents of working memory, so that for information to influence behavior, it must be in working memory.

Semantic memory contains the agent's long-term declarative knowledge, such as facts about the world, and corresponds to ACT-R's long-term declarative memory. Information can be retrieved from semantic memory into working memory via a query. A query is initiated using a *cue* that is composed of a single-level symbolic directed graph, anchored in a single node. Consider an example where there has previously been a retrieval for the word "activation" that returned a result with substructure `^meaning A1437`. The agent may then decide to retrieve a second word sense of "activation" using the following cue: (`<cue>^word-string activation ^meaning A1437 -`), where "-" is used to prohibit the retrieval of the previous word sense. All nodes in semantic memory that match the cue are found. If no node matches, then the query fails. If more than one node matches, a *bias term* is computed for every cue matching node and the node with the highest bias term is the result. One important component of the bias term is base-level activation (BLA). BLA combines information on the recency and frequency of

a node's previous accesses. Although BLA is useful, it does not support context-sensitive retrieval, where structures in the context (the contents of working memory) influence the bias term. One approach to incorporating context into the bias term is to use spreading activation (SA) as another component. Adding together the BLA component bla_m and the SA component sa_m for every cue matching node m gives us an overall bias term BT_m.

Naive Spreading Activation

Activation spreads out from semantic memory nodes that are in working memory to adjoining nodes in semantic memory. One point of variability is whether activation spreads in the direction of edges (forward), opposite that direction (backward), or in both directions. Our algorithms are agnostic on spread direction and support all three directions. For simplicity and ease of analysis, our implementations track and record the SA- and BLA- component independently. Their only interaction is during the calculation of the overall bias term, simply related: $bla_m + sa_m = BT_m$

For the Naive Algorithm, spreading begins whenever a node n enters the context and becomes a *source* of spread. In Figure 1, node **A** is the source and spreads activation of .45 forward to the two nodes labeled **B** and **C**. Nodes that receive activation are spread *recipients*, and they can accumulate activation from many sources or even from the same source at varying depths, such as nodes **E** and **F**. The total accumulated activation for recipient r, is denoted as s_T.

The calculation of the activation of a recipient depends on three factors. The first is the initial activation of the source, which we set to 1. Second, as activation spreads deeper, there is a decay factor, $p < 1$. In this example, the decay factor is .9. Third, the activation of a parent node is divided equally among all of its children nodes, leading to the fan effect through the spread of activation (Anderson, 1983b). Thus, the calculation for the activation of a recipient node r, where there are a total of k children nodes from parent node s_n is $s_r = \frac{1}{k} \times p \times s_n$. For the children of **A** (nodes **B** and **C**), the calculation is $\frac{1}{2} \times .9 \times 1 = .45$. If a recipient receives activation from two distinct parents, the activations from both parents are summed together. Thus, for node **F**, **C** and **G** are both parents that respectively issue .45 and .135 to result in an activation value of .19575 for **F**.

We denote the activation that accumulates in a spread recipient, r, from a source, c, as $s_{r,n}$, so that the total activation for node $s_T = \sum_{c \in C} s_{r,c}$ where C is the set of all context nodes that are sources. If a node does not receive any activation, then $s_T = 0$.

The total spread from a source can be controlled in multiple ways. For example, a spread can be restricted to a fixed distance from the source node, called the *depth limit* or there can be a limit to the total number of nodes traversed, termed the *spreading size limit*. In our experiments, for simplicity we use a fixed spreading size limit, which is 300. The spreading size limit is applied within the context of a breadth-first traversal of the graph, so that activation is spread to all nodes at the current depth before it is spread to nodes at the next depth level. With a size limit, it is possible to not spread to every node at the same depth. For example, if we use a size limit of 300 and we assume a regular graph where every node has exactly five children, after depth 1, 5 nodes will be visited; after depth 2, $5+25=30$ nodes will be visited; and after depth 3, $5+25+125=155$ nodes will be visited. An addition 145 nodes at depth 4 (out of 625) will be explored, but 480 nodes at depth 4 will not.

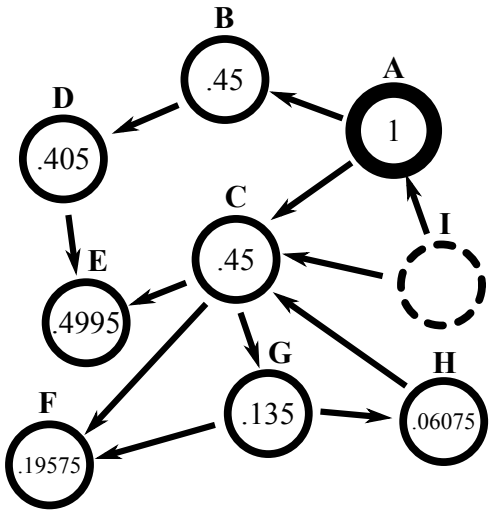


Figure 1: An example network, with forward spread from a single source **A**. Initial activation is 1, decay factor is .9, and depth limit is 3.

Observations on Naive Spreading Activation

In order to compute the overall bias term, a naive algorithm spreads from every context node when it is introduced into working memory and updates the SA component of every recipient per each spread. In such an implementation, the primary cost of spreading activation is the breadth-first traversal that computes the activation for nodes that receive spread from a given source. A secondary cost is updating the new value of the SA component and recalculating the overall bias term, which for the naive algorithm is performed for every spread recipient. In a naive algorithm, these costs scale with the number of source nodes multiplied by the spread size limit.

In the following sections, we identify properties of processing that suggest ways of reducing the costs of a naive algorithm. Many of these shortcuts are obvious, while others are more subtle, but in general the unnecessary computation can be avoided by one major approach: avoid calculations of spread until it is necessary, because it is possible they may never be necessary. This approach is called *lazy evaluation* and attempts to only compute activations that play a role in determining which candidate is retrieved from memory. In the naive algorithm, a significant proportion of nodes that re-

ceive spread do not influence which candidate is retrieved from memory. It is because of this that lazy evaluation can reduce computation while resulting in no change to which candidates are retrieved.

Our final algorithm incorporates optimizations made possible by the following observations.

There are Consistent Elements Shared Between Old and New Context

If there are no changes in the set of context elements, the results of spread will be exactly the same on subsequent cycles. Thus, spread only has to be computed when context elements change, which we call **Change-Only Processing**. This optimization is expected to have a large impact because working memory remains largely unchanged from one cycle to another. For an architecture where working memory has rapid changes, this may not be much of an improvement.

The Long-Term Memory Network Changes Slowly

The knowledge in semantic memory is not expected to change rapidly. Thus, the breadth-first traversal that computes the spread is relatively stable. We can take advantage of this stability by explicitly caching a trace of the breadth-first traversals and the activation values they produce. These can be used to directly access these values for updating the SA component without performing the breadth-first traversals. Whenever an edge is added to or removed from semantic memory, the traversals containing the parent of that edge are no longer valid. Under these circumstances, if these traversals are needed in the future, they will be recomputed. This improvement is simply referred to as **Caching**.

One implication is that we can compute traversals during task initialization. Thus, if the agent starts with a large knowledge base pre-loaded into its long-term semantic memory, it can pre-compute and cache the traversals for future use. Nevertheless, the agent must still recalculate traversals when the network changes according to the invalidation cases we outlined above. This is **Precalculation**.

Queries are Less Frequent than Context Changes

Even when there are context changes during a cycle, spread does not need to be computed unless there is a query. Even when the calculation of spread is tied to only changes in the context, this modification eliminates calculations for cases where context elements become active and then become inactive without any intervening queries. This improvement is called **Query-Deferred Calculation** and exists in ACT-R. This improvement also supports implementation of two further improvements.

A Query's Cue can be Unambiguous

If a cue is used that is constraining enough so that only one long-term memory node matches, there is no need to spread. We call this improvement **Ambiguity-Only Processing**.

The Number of Cue Matches is Small

A naive approach to spreading activation normally computes the activation of all recipients of a spread. However, it is rare that more than a small percentage of those nodes match the cue, and the SA component is needed only for those nodes that match. Thus, if a node does not match the cue, it is unnecessary to calculate the overall bias term for that node. Even when only a few constraints are included in a cue, they can eliminate a substantial proportion of the nodes from consideration.

In response to this observation, our approach flips the normal way of thinking about computing spread. Instead of updating the SA component of every recipient, our algorithm only computes the overall bias term for nodes that match the cue. This improvement is named **Candidate-Only Processing** and it is included in ACT-R. Note that if a cue has so little constraint such that every node is a candidate, this optimization will not help.

While this series of improvements generates our final algorithm, we restate the final algorithm explicitly below.

Algorithm Review

We reconceptualize spreading activation as the calculation required to provide the bias term necessary for context-sensitive retrieval. In algorithm 1, we follow the procedure PROCESSAGENTCYCLE() every cycle. The first step is to check whether or not a query is present. If not, processing stops. This corresponds to our **Query-Deferred Calculation** improvement. If a query is present, then there is a check as to whether there is only one node that matches the cue. If so, all spreading activation calculation is skipped and that node is returned. This is called **Ambiguity-Only Processing**. However, if the cue is ambiguous, then spreading activation is computed using DOTRAVERSALS() and DOAPPLICATIONS().

In DOTRAVERSALS(), if there are changes to the context elements then the following processing occurs. If a source's traversal has never been calculated or there has been a change to the network that invalidates the traversal, then it is necessary to recalculate the traversal via breadth-first search (TRAVERSE()). A map from node to traversal, *cachedSpread*, maintains a history of currently usable traversals. If these conditions do not hold, then a cached traversal is retrieved to bypass additional calculation.

After all source node traversals are calculated and retrieved, DOAPPLICATIONS() updates the SA component of all the recipients in these traversals that also match the cue as looked up against a recorded history of currently usable traversals. Updating the SA component of only the cue-matched nodes corresponds to **Candidate-Only Processing**.

Finally, the cue-matched node with the highest overall bias term is returned as the result to the query.

The worst case for this algorithm is when there are frequent changes to declarative memory that invalidate the cached traversals, and when there are frequent changes to context elements that require continual recalculation of the traversals. In

Algorithm 1 : Lazy algorithm for spreading activation

```
cachedSpread                                   ▷ global variable

1: function DOTRAVERSALS(contextChanges)
2:     for source ∈ contextChanges do
3:         if source ∉ cachedSpread OR
           ISINVALID(cachedSpread[source]) then
4:             spread ← TRAVERSE(source)
5:             cachedSpread[source] ← spread

1: function DOAPPLIES(cueMatches, contextChanges)
2:     for match ∈ cueMatches do
3:         if match ∈ cachedSpread[contextChanges] then
4:             UPDATEBIASTERMOF(match)

1: procedure PROCESSAGENTCYCLE()
2:     if agent issues a query command then
3:         cueMatches ← DOQUERY(cue)
4:         if SIZEOF(cueMatches) == 1 then
5:             ADDTOWMEM(match)
6:         else
7:             DOTRAVERSALS(contextChanges)
8:             DOAPPLIES(cueMatches, contextChanges)
9:             contextChanges ← ∅
10:        ADDTOWMEM(MAX(cueMatches))
```

this worst case, our algorithm essentially performs the naive algorithm.

Evaluation

Our hypothesis is that our proposed algorithm can result in a significant reduction in the time spent computing spreading activation and that each component of the algorithm provides some benefit. To test these claims, we evaluate the time efficiency as we incrementally incorporate each component of the final algorithm.

The task we use is the word sense disambiguation (WSD) task that we previously used for evaluating implementations of base-level activation (Derbinsky & Laird, 2011). In this task, the agent must disambiguate the word senses used in a sentence. Each input word is annotated with its name and part-of-speech (e.g. noun) but not its sense. When an agent encounters and issues a query for the word, such as the word "English" with part-of speech "noun", it must choose one of the following possible senses: 1) the West Germanic language; 2) the humanities discipline; 3) the people of England; 4) the spin given to a ball in pool or billiards. The agent keeps retrieving senses until the retrieved sense matches the correct sense.

The test sentences and the ground truth are provided by SemCor, a popularly used sense-tagged corpus. SemCor consists of 352 texts from the Brown corpus (Kucera & Francis, 1967), with every word linked to its correct sense in the English lexical database WordNet, version 3.0 (Miller, 1995). Our construction of WordNet 3.0 includes all synset and lemma links for every part-of-speech, and our construction of SemCor includes all available sense-tagged words, numbering 217,918, of which approximately 75% are multisense[1].

In our experiment, there are seven different agents, corresponding to different spreading activation algorithms. These are listed in Table 1. All agents are preloaded with our construction of WordNet 3.0 in their semantic memory and they all use Soar's existing base-level activation mechanism in addition to spreading activation. We compare as well to one agent that does not use spreading activation, instead using only base-level activation.

All agents iterate through all SemCor sentences, maintaining in working memory the retrieved correct word sense for all words previously encountered within a paragraph as context. Table 1 displays the time spent on spreading activation during the task. The execution of the task is deterministic with negligible variance in execution times. All spreading activation agents have a spread size limit of 300, and they all compute exactly the same spread values (and bias terms) for all the candidate retrievals, and they retrieve the same node from semantic memory. Thus, the seven algorithms differ only in the efficiency of computing the retrieved nodes.

All agents ran for a total of 1,644,058 decision cycles while issuing a total of 565,223 queries. The naive algorithm, omitting all improvements, took over 100,000 seconds. Every change to the algorithm decreased execution time. The amount of time our final algorithm took on spreading activation alone was 245 seconds. On average, the amount of time spent on the rest of the agent's processing was 290 seconds (not shown in Table 1). When examined at the individual query level, the final algorithm spent an average of 0.43ms per query on spreading activation compared compared to 5.87ms for the naive algorithm with change-only processing.

We confirm that precalculation (and thus the corresponding reduction in the breadth-first traversals during the task) speeds up the agent. The memory cost to precalculation is storage of the traversal to 300 nodes for each node. While query-deferred processing has little direct impact, it supports candidate-only processing and ambiguity-only processing. While the effect of ambiguity-only processing is modest, candidate-only processing shows a significant improvement associated with selectively updating only spread recipients that are potential query results. The naive algorithm, which omits all improvements, is the slowest.

The amount of time to calculate spread from a single node is expected to scale linearly with spreading size limit. As a check, we used a test agent that first randomly selects a word and then adds new word information to the network. The randomly-selected word serves as a context element. The agent then initiates an artificially constrained query, such that

[1]The SemCor and WordNet 3.0 data sets are available to download at `http://web.eecs.umich.edu/~mihalcea/downloads.html#semcor` and `http://wordnet.princeton.edu`, respectively.

Spreading Activation Mechanism	Spread Time (s)	Spread Time Per Query (ms)
Naive Algorithm	$> 100{,}000$	
+ Change-Only Processing	3,316	5.87
+ Caching	1,200	2.12
+ Precalculation	810	1.43
+ Query-Deferred Processing	803	1.42
+ Ambiguity-Only Processing	778	1.38
+ Candidate-Only Processing	245	.43

Table 1: Timed performance on the WSD task across seven spreading activation variants. Rows with prefaced with "+" denote incremental cumulative improvements to our algorithm.

the breadth-first traversal must be calculated for the new context element and that query has a sufficiently general cue such that the candidate set includes all spread recipients. The test agent thus induces the maximum cost of a single network change. As expected, the maximum spreading times shown in this figure are significantly greater than the average times achieved in the WSD task.

The timing results for this test are found in Figure 2. Graph points that fall below the linear trend reflect spread traversals that exhaust the network before reaching the spread size limit. We note that given our random selection of words, there is some noise and furthermore that a traversal of a given size can have variable cost depending on whether repetition in the traversal reduces the number of elements requiring application further below the spread size limit. However, it is overwhelmingly the spread size limit that determines the total cost and we observe the expected linear scaling.

While the termination criterion of spreading size limit allows for direct control over computational cost and is convenient for the above analysis, we add an additional termination criterion. A spreading size limit of 300 does not provide a meaningful bound in terms of the influence spreading has on retrieval. In the presence of noise or uncertainty, small values of spreading activation may be irrelevant. We thus introduce a threshold termination criterion such that spreading traversals terminate if spreading activation values generated in the traversal are below the threshold. In other words, we assume a minimum acceptable spreading activation value as an adjustable parameter. We also change the traversal so that instead of breadth-first traversal, the traversal is biased to where there is still the most spread to distribute.

The intuition is to pick a value such that spreading activation is not applied if it would be "lost in the noise." Note that in ACT-R, such a noise term is added to activation. Consider an ACT-R noise set to .1. A threshold of .0025 in Figure 3 represents a 95% chance that such a noise magnitude is larger than the terminated spread. Figure 3 shows that such a termination criterion would result in spread sizes of approximately 65 nodes. Per query, a spread size of 65 nodes is expected to take an average of .094ms. The threshold has the potential to

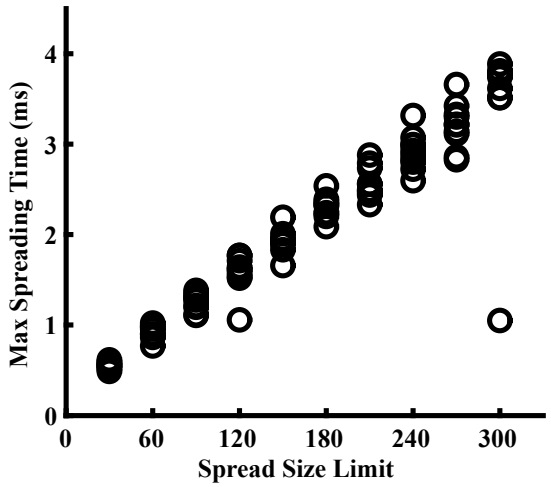


Figure 2: The maximum time spent on spreading activation from a single context node, with varying spread size limit, for randomly selected words.

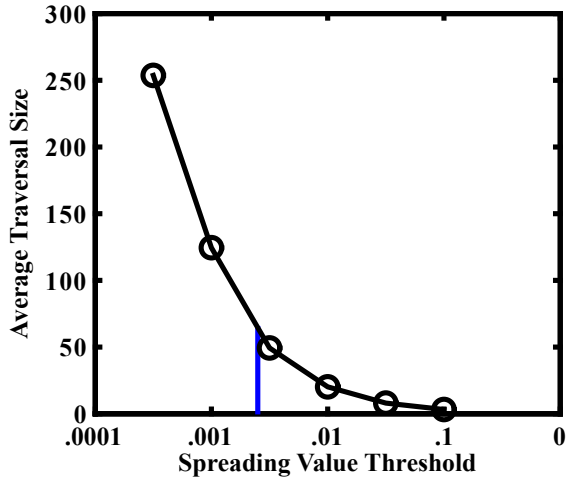


Figure 3: The average sizes of spreading traversals are plotted with varying thresholds for termination. The vertical line denotes a threshold value of .0025.

change which candidates are retrieved.

Conclusion and Future Work

A central motivation of implementing spreading activation is to support context-sensitive retrieval for cognitive agents. To satisfy the constraints of a cognitive architecture while meeting the demands of complex, dynamic, or real-world environments, spreading activation must be efficient and reactive. We have developed an optimized algorithm for spreading activation that has an average time of .43ms for a spread to 300 nodes. Although optimized, there is no compromise in correctness – results of the spread are *exactly the same* as the results of a straightforward (naive) algorithm. Adding a threshold-based termination criterion for spread based on noise or confidence further reduces the cost to .094ms, albeit potentially changing query results. We expect that such efficient spreading activation will change how spreading is used in cognitive architectures. It will be possible to explore deeper spreads where there are more indirect associations between concepts, and it will be possible to use it for real-world applications.

In the future, we plan to further evaluate this algorithm on much larger networks and networks with more varied structure to get a better profile of its performance characteristics for different network organization and dynamics.

We also plan to extend our algorithm so that it includes a temporal decay for spreading activation. Our plan is to initialize the magnitude of the spread from a source node with that source node's base-level activation. Additionally, we plan to extend the representation of semantic memory so that it includes association strengths between nodes. These two changes should have only minimal impact on the spreading algorithm and its efficiency while allowing us to study algorithms that dynamically modify those association strengths based on the co-occurrence of nodes in working memory. This suite of changes has the potential to allow spreading activation to adapt to an agent's experience, which is lacking in our current implementation.

Acknowledgments

The work described here was supported by the Office of Naval Research under grant number N00014-08-1-0099. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the ONR or the U.S. Government.

References

Anderson, J. R. (1983a). The architecture of cognition. *Cambridge, Mass.: Harvard University Pres*.

Anderson, J. R. (1983b). A spreading activation theory of memory. *Journal of Verbal Learning & Verbal Behavior*.

Chen, Y., Petrovic, M., & Clark, M. (2014). Semmemdb: In-database knowledge activation. In *Flairs conference.*

Derbinsky, N., & Laird, J. E. (2011). A functional analysis of historical memory retrieval bias in the word sense disambiguation task. *Ann Arbor*, *1001*, 48109–2121.

Derbinsky, N., Laird, J. E., & Smith, B. (2010). Towards efficiently supporting large symbolic declarative memories. In *Proceedings of the 10th international conference on cognitive modeling* (pp. 49–54).

Douglass, S. A., & Myers, C. W. (2010). Concurrent knowledge activation calculation in large declarative memories. In *Proceedings of the 10th international conference on cognitive modeling* (pp. 55–60).

Edmonds, M., Atahary, T., Taha, T., & Douglass, S. A. (2015). High performance declarative memory systems through mapreduce. In *Software engineering, artificial intelligence, networking and parallel/distributed computing (snpd), 2015 16th ieee/acis international conference on* (pp. 1–8).

Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern / many object pattern match problem. *Artificial Intelligence*, *19*(1), 17–38.

Kucera, H., & Francis, W. N. (1967). *Computational analysis of present-day american english*. Providence, RI: Brown University Press.

Laird, J. E. (2012). *The Soar cognitive architecture*. MIT Press.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, *28*, 29–41.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Toward a Unified Theory of Learned Trust

Ion Juvina[1] (ion.juvina@wright.edu), Michael Collins[1,2] (collins.283@wright.edu), Othalia Larue[1] (othalia.larue@wright.edu), & Celso de Melo[3] (demelo@usc.edu)
[1]Department of Psychology, Wright State University, 3640 Colonel Glenn Hwy., Dayton, OH 45435 USA
[2]Air Force Research Laboratory, Dayton, OH 45435 USA
[3]Institute for Creating Technologies, University of Southern California, 12015 Waterfront Dr., Playa Vista, CA 90094 USA

Abstract

A proposal for a unified theory of learned trust is presented. A number of limitations of a published computational cognitive model of learned trust are discussed. A solution is proposed to overcome these limitations and expand the model's scope of applicability. The revised model integrates several seemingly unrelated categories of findings from the literature and makes unintuitive predictions for future studies. The implications of the model for the advancement of the theory on trust are discussed.

Keywords: trust; trustworthiness; trust propensity; learned trust; computational cognitive model; unified theories

Introduction and Background

Newell (1990) called for unified theories of cognition specified computationally as cognitive architectures. A cognitive architecture is a single system of cognitive mechanisms that operate together to produce the full range of human cognition. Unified theories are the quintessence of scientific progress. They constrain the myriad of possible interpretations of empirical data, facilitate communication among theorists, and motivate new avenues for empirical research. Here we focus on the field of trust research, particularly on what has been referred to as learned trust (Hoff & Bashir, 2015), and attempt to integrate it in the ACT-R cognitive architecture (Anderson, 2007). Although the field of trust already comprises an impressive volume of empirical findings, micro-theories, meta-analyses, and integrative accounts (e.g., Rousseau, Sitkin, Burt, & Camerer, 1998; Mayer, Davis, & Schoorman, 1995; Schoorman, Mayer, & Davis, 2007; Lee & See, 2004; Hoff & Bashir, 2015; Schaefer, Chen, Szalma, & Hancock, 2016), it could benefit from the kind of integration that is afforded within a cognitive architecture. Studying trust from a cognitive architecture perspective allows not only integration of various empirical findings from the trust literature but also understanding how trust relates to other cognitive mechanisms and phenomena.

The starting point for the effort reported here is a published model of learned trust (Juvina, Lebiere, & Gonzalez, 2015; referred to as "the published model" in the remainder of the paper). In the next section we briefly review the key features of the published model and discuss its main strengths and limitations. Then, we devote another section to a revised model (referred to as "the revised model" in the remainder of the paper) that is intended to overcome the limitations of the published model. Subsequently, we show that the revised model can account for a number of results from the trust literature. In the last section, we discuss possible ways to further improve the revised model and suggest that it has the potential to integrate a wide range of empirical findings and thus it can inform the development of a unified theory of learned trust.

Critique of the Published Model

The published model (Juvina et al., 2015[1]) was built in the ACT-R architecture and was intended to account for learning within and between two games of strategic interaction – Prisoner's Dilemma (PD) and Chicken Game (CG). The model is not hardwired to play a particular game; it learns to play any 2X2 game (Rapoport, Guyer, & Gordon, 1976) based on the payoff matrix that it experiences as it plays. Initial attempts to account for the transfer of learning effects between the two games in both directions (PD-CG and CG-PD) observed in the human data (Juvina, Saleem, Martin, Gonzalez, & Lebiere, 2013) based solely on the existing learning mechanisms of the ACT-R architecture were unsuccessful. A novel trust learning mechanism had to be added to the model to account for all the learning and transfer of learning effects in the data. Essentially, this trust mechanism allows models to learn not only about the task at hand but also about other models with which they interact. Although learning in individual settings has been extensively studied, learning about others has not received much attention in the cognitive modeling field. It is not clear whether learning about other agents uses the same cognitive mechanisms as learning about inanimate entities. Yet, empirical evidence suggests that learning from others and learning about others can influence task specific learning (Biele, Rieskamp, & Gonzalez, 2009; Yaniv & Kleinberger, 2000; Harris & Corriveau, 2011). The published model uses instance-based learning (Gonzalez, Lerch, & Lebiere, 2003) for opponent modeling and reinforcement learning for action selection. In addition, the reward changes as the game unfolds depending on the dynamics of the interaction between the two models. The players learn to trust each other and this affects their reward structure and subsequently their strategies. The trust mechanism consists of a "trust accumulator" that represents the perceived trustworthiness of the other model and a "trust-invest accumulator" that represents the perceived necessity to develop trust – a characteristic of the situation. For example, when the two models find themselves in a

[1] Model code available at: http://psych-scholar.wright.edu/astecca/software

self-reinforcing cycle of mutual defection, the perceived necessity to develop trust increases. This was a necessary addition to the model to overcome situations in which both players strongly distrust each other and persist in choosing a mutually destructive outcome. Humans are able to identify and (sometimes) overcome those situations.

The two accumulators (trust and trust-invest) are used to determine the dynamics of the reward structure. Each accumulator starts at zero. When they both are less than or equal to zero, the model will act selfishly by trying to maximize the difference between their own payoff and the opponent's payoff. This quickly leads to the mutually destructive outcome continually occurring during the game, which decreases trust in the other model but increases the model's perception of trust necessity. Once the latter is positive, a model acts selflessly, trying to maximize the opponent's payoff. This can lead to mutual cooperation and development of trust or models may relapse into a mutual destructive choice. When the trust accumulator is positive, a player tries to maximize joint payoff and avoid exploitation. Thus, the model switches between three reward functions depending on the dynamics of trust between the two players.

Strengths of the published model

The main contribution of the published model was to show that trust learning interacts with task specific learning to account for a range of learning effects in the human data. This model has the potential to inform a unified theory of learned trust because it is implemented in a cognitive architecture and it specifies how various learning mechanisms interact with (and constrain) each other. In agreement with the literature on trust, the published model's trust is learned as a function of perceived trustworthiness (Mayer et al., 1995; Hoff & Bashir, 2015). In addition, the published model suggests that a player's learned trust also depends on perceived trust necessity, which is in and of itself an important contribution to the literature. A validation study based on predictions of the published model showed that both perceived trustworthiness and perceived trust necessity are important antecedents of trust formation (Collins, Juvina, & Gluck, 2016).

Limitations of the published model

Most of the limitations of the published model stem from the fact the model was initially not intended to be comprehensive model of learned trust. Instead, the model had to learn trust in order to account for transfer of learning effects observed in the human data. The published model assumes that trust starts at zero and only the trust developed during the interaction between the two players matters. However, there is overwhelming evidence that a player may trust another player even in the absence of any interaction between the two players (McKnight, Cummings, & Chervany, 1998) and this initial propensity to trust determines to some extent the trust that develops during the interaction (Berg, Dickhaut, & McCabe, 1995; Dirks & Ferrin, 2001). In addition, trust propensity may be (at least in part) the result of learning that occurred prior to the current interaction (Collins et al., 2016) and a comprehensive model of learned trust should not ignore prior learning, particularly because prior learning may interact with current learning. This aspect was not relevant in the published model because the model interacted with only one other model, but it becomes very relevant in the context of learning from interacting with multiple agents in sequence and transfer of learning from one agent to another (see the black-hat-white-hat effect in the next section).

The published model's learning equation is a linear function that increases with every instance of evidence of trustworthiness and decreases with every instance of evidence of untrustworthiness (and similarly for evidence of trust necessity). The rate of accumulation is equal for positive and negative evidence and is constant throughout the entire history of interaction. The following is the equation for state trust learning that was used in the published model,

$$ST_t = ST_{t-1} + PET_t \qquad (1)$$

where ST_t is state trust at time t, ST_{t-1} is state trust at time t-1, and PET_t is perceived evidence of trustworthiness at time t. A similar equation was used for trust necessity.

This equation worked well in the context of the published model but is problematic because it is not in full agreement with what is known about the dynamics of trust. Trust is hard to gain and easy to lose, a characteristic that has been referred to as trust asymmetry (Slovic, 1993). Trust learners exhibit the same negativity bias that is described in the impression formation literature (Skowronski & Carlston, 1989; Yaniv & Kleinberger, 2000), that is, unfavorable information tends to be more influential than favorable information. In addition, early evidence has a stronger impact on trust formation than late evidence (Lount, Zhong, Sivanathan, & Murnighan, 2008). In general, learning equations tend to be power functions (Newell & Rosenbloom, 1981; Anderson, 2007) and it would be surprising if trust learning were an exception.

Another limitation of the published model is that it assumes that all trustors are able to assess equally well trustworthiness and trust necessity. However, a trustor's cognitive ability to assess a trustee's trustworthiness has been proposed to be an important antecedent of trust (Lyons, Stokes, & Schneider, 2011; Sturgis, Read, & Allum, 2010; Yamagishi, Kikuchi, & Kosugi, 1999). In general, cognitive ability is an important predictor of learning, thus it is not surprising that it is also related to learned trust.

The Revised Model

Before introducing our revisions to the published model, we specify the terminology used in this model. *Trait trust* is the term we use for trust propensity (also called dispositional trust in the literature). *State trust* is the trust that develops during a particular interaction in a particular situation, thus, is a function of the *perceived evidence of trustworthiness*

and *perceived evidence of trust necessity*. In our view, *learned trust* includes both trait and state trust; trait trust is learned from the ensemble of past interactions and state trust is learned from the current interaction. The starting value of state trust at the beginning of the current interaction is the trustor's trait trust. This reflects the finding that humans place a certain amount of trust in strangers that they know nothing about (Berg, Dickhaut, & McCabe, 1995). State trust is updated during an interaction depending on perceived evidence of trustworthiness and perceived evidence of trust necessity. At the end of the current (repeated) interaction, trait trust is updated with an increment that is a function of the state trust developed in the current (just ended) interaction. This reflects the finding that trait trust changes as a function of experience (Collins et al., 2016). *Trait trust deviation* is the difference between the trait trust value at the end of the current interaction and the trait trust value at the beginning of the interaction. The trustor's *cognitive ability* is indicated by the accuracy of the trustor's judgments of trustworthiness and trust necessity.

The revision[2] of the published model consists of replacing the linear function that was used to update the trustor's state trust with the following power function,

$$ST_t = ST_{t-1}^{a} + PET_t - b * TTD \qquad (2)$$

where ST_t is state trust at time t, ST_{t-1} is state trust at time t-1, a is a constant power exponent with a value less than 1 (a<1), PET_t is perceived evidence of trustworthiness at time t, TTD is the trait trust deviation computed after the previous interaction with another person, and b is the perception bias that scales how much PET_t is adjusted as a function of the trustor's previous experience with another trustee. A similar equation was used for trust necessity.

In the revised model, both trait and state trust are positive or zero. A value of zero signifies the absence of trust. The evidence of trustworthiness can be positive (indicating a degree of trustworthiness) or negative (indicating a degree of untrustworthiness). The initial value of state trust is the value of trait trust that was updated after the previous interaction with another person (ST_{t0} = TT). In our simulations, we set the initial trait trust somewhere in the middle of the range of values that state trust can take during a repeated interaction with a specific person, depending on the range of values that the evidence of trustworthiness can take. We assume that weighting of the evidence is task specific.

The continuous value of state trust can be used to make categorical judgments (i.e., trust or distrust) by comparing it against the value of trait trust. If the current value of state trust is greater than the value of trait trust, then the trustor is said to trust the trustee. If the current value of state trust is less than the value of trait trust, then the trustor is said to distrust the trustee.

[2] Model code available at: http://psych-scholar.wright.edu/astecca/software

The power exponent (a) is currently set to 0.99 in all our simulations. The assumption behind this component of the equation is that the more recent values are more important than the older values of state trust. A consequence of this assumption is that trust decays in time if new evidence of trustworthiness is not perceived. Note that for a = 1 and TTD = 0, equations (1) and (2) are identical.

Figure 1 shows a hypothetical case in which a trustor repeatedly interacts with a trustee for 200 rounds. The trustor perceives evidence of trustworthiness (PET = 1) for the first 100 rounds, then evidence of untrustworthiness (PET = -1) for 5 rounds, and then again evidence of trustworthiness (PET = 1) for the remaining 95 rounds. State trust accumulates rapidly in the first 50 rounds after which it starts to approach an asymptote, that is, a state of diminishing returns for every new piece of evidence of trustworthiness. In addition, the state trust that was accumulated over 100 rounds is lost almost entirely in 5 rounds, manifesting trust asymmetry (Slovic, 1993).

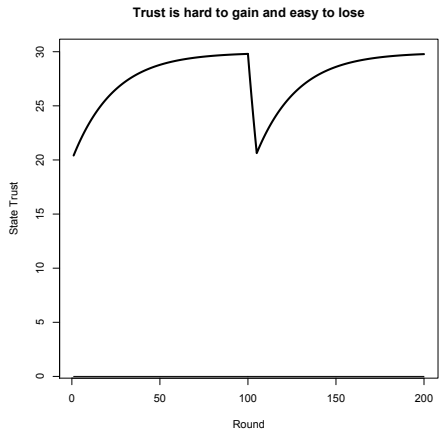


Figure 1: A hypothetical case illustrating how state trust changes over the course of 200 rounds of interaction with another player. The trustor perceives evidence of trustworthiness for the first 100 rounds, then evidence of untrustworthiness for 5 rounds, and again evidence of trustworthiness for 95 rounds.

The term trait trust deviation (TTD in equation 2) becomes relevant when a trustor interacts with multiple trustees in sequence. In such cases, empirical studies suggest that the experience from a previous interaction influences how the trustor perceives the evidence of trustworthiness in the current interaction. For example, De Melo, Carnevale, and Gratch (2011) review evidence and possible explanations for the black-hat/white-hat (or bad-cop/good-cop) effect from the negotiation literature: playing a first game with an opponent with a competitive stance (black-hat) followed by a second game with an opponent with a cooperative stance (white-hat) is more effective in reducing distance to agreement than any other pairing of the black-hat and white-hat opponents (Hilty & Carnevale, 1993). We implemented the explanation of the black-hat/white-hat effect that is based on the concepts of adaptation and comparison level

(Helson, 1964). Theories of adaptation propose that people become accustomed to a reference point as a result of prior experience; this point then serves as a comparison for the judgment of subsequent experiences. Thus, a cooperative second bargainer should be judged as more cooperative if the first bargainer was competitive rather than cooperative. In terms of our learned trust theory, the prior experience of untrustworthiness shifted the trustor's reference point toward low values of trustworthiness. In this context, evidence of trustworthiness from a new interaction is perceived as outside of the expected range which gives it a larger subjective weight. In our model, we assume that the change in the subjective perception of the new evidence is proportional to the adjustment (i.e., adaptation in Helson's terms) of the reference point caused by the previous experience. The reference point is the trustor's trait trust. For example, if the trustor's previous experience with an untrustworthy trustee caused a large shift in her trait trust, the corresponding bias in her subjective perception of a new trustee will also be large (and vice-versa). Thus, a trustor's previous trait trust deviation (TTD) determines the extent to which the perceived evidence of trustworthiness (PET) is adjusted.

To conclude the description of the revised model, only the trust learning mechanism has been revised, all the other mechanisms of the published model (learning to anticipate the opponent's move and to select the best move in each context, see Juvina et al., 2015) have been left unchanged.

Model Validation

We expect that the revised model is able to generalize to a wide range of empirical phenomena while maintaining the ability of the published model to explain the learning and transfer of learning effects from the original dataset.

Learning and transfer of learning effects in Prisoner's Dilemma and Chicken Game

Juvina et al. (2013) recruited 120 participants to play Prisoner's Dilemma and Chicken Game for 200 rounds each. The participants were paired with one another and assigned to play the two games in one of two order conditions: PD-CG and CG-PD. The results revealed a wide range of within-game learning and between-game transfer of learning effects. The published model was fit in its entirety to this dataset by tweaking 11 free parameters (see Table 4 in Juvina et al., 2015). With regard to the revised model, only the six free parameters associated with the trust mechanism were refit to the human data reported in Juvina et al. (2013). Four of the six parameters are associated with the "trust accumulator" that represents the perceived trustworthiness of the other player and the other two are associated with the "trust-invest accumulator" that represents the perceived necessity to develop trust. The values of these parameters specify how much perceived evidence of trustworthiness (PET in equations 1 and 2) is added to (or subtracted from) the trust accumulator for each outcome of the game. Two of the six parameters (i.e., the parameter with the lowest absolute value for each accumulator) were kept at their values from the published model, thus, allowing only four model parameters to fluctuate. The fit procedure maximized the correlation (r) and minimized the root mean squared deviation (RMSD) between the model data and the human data[3].

Table 1 shows the best fitting parameter values for the revised model and the published model. They did not change dramatically; as a matter of fact, one of them did not change at all, even though it was allowed to vary freely. Thus, only three parameters have been readjusted in the revised model. These parameters were held constant for all but one of the simulations reported below. They were readjusted for Lount et al. (2008) data because a very different payoff matrix was used in that study.

Table 1. The best fitting parameter values for the revised model and the published model for each of the four game outcomes, mutual cooperation (CC), unilateral cooperation (CD), unilateral defection (DC), and mutual defection (DD). An asterisk (*) indicates that a particular value was held constant during the model fitting procedure.

Outcome	Published model		Revised model	
	Trust	Invest	Trust	Invest
CC	3	NA	6	NA
CD	-10	-1	-7	-1
DC	10	NA	9	NA
DD	-1	.18	-1*	.18*

The fit of the revised model to the human data (r = .90, $RMSD$ = .07) was slightly (but not significantly) better than the fit of the published model (r = .89, $RMSD$ = .09). The revised model also exhibited the same transfer of learning effects observed in the human data.

Collins et al. (2016) conducted a follow-up study in which 320 participants recruited from the website Amazon Mechanical Turk played PD and CG for 50 rounds each in one of four possible game orders (PD-PD, PD-CG, CG-PD, or CG-CG). Participants were paired with computerized confederate agents whose behavior (i.e., strategy & trustworthiness) was manipulated to result in 16 different experimental conditions. The published model (Juvina et al., 2015) was used to generate *a priori* predictions for Collins et al. (2016) study. The predictions were published before the data were collected (Collins, Juvina, Douglas, & Gluck, 2015). A majority of the model predictions across all of the sixteen experimental conditions was confirmed and the trust mechanism was proven to be a necessary component of the published model (see Collins et al., 2016, for details). Here we test the revised model against the dataset from Collins et al. (2016) without any parameter tweaking. The data includes round-by-round proportions for five outcomes in

[3] High performance computing facilities at the Air Force Research Laboratory and the web service mindmodeling.org (Harris, 2008) were used for the model fitting procedure.

16 conditions. The revised model accounts for the human data slightly (but not significantly) better (r = .68, *RMSD* = .33) than the published model (r = .64, *RMSD* = .33).

Unified account of trust and distrust effects

It has been proposed that trust and distrust are different constructs (Lewicki, McAllister, & Bies, 1998; Sitkin & Roth, 1993). Here we suggest that the different dynamics of trust and distrust can be modeled by a single equation. In the previous section we showed how equation 2 produces trust asymmetry (Slovic, 1993; see Figure 1). A consequence of trust asymmetry is the fact that early trust breaches are more influential than late trust breaches for the overall trust that develops in a repeated interaction, which is exactly what Lount et al. (2008) found. Lount et al. (2008) conducted two experiments in which participants played an iterated game of Prisoner's Dilemma for 30 rounds. Participants were assigned to 1 of 4 experimental conditions (control, immediate, early, and late) and played the game with a confederate agent whom they were told was another participant. During the control condition, the confederate agent cooperated on all 30 rounds. In the other three conditions, the confederate agent cooperated on each round except for two consecutive trials on which it defected. These trust breaches occurred immediately (rounds 1 and 2), early (rounds 6 and 7), or late (rounds 11 and 12). The main finding revealed that the immediate and early breaches significantly decreased the frequency of cooperation during the last ten rounds of the game as compared to the late breach.

Our revised model is able to account for the basic pattern of results, that is, the different amounts of cooperation in control, immediate, early, and late conditions (r = 0.99, *RMSD* = 0.33). One possible explanation for the large RMSD is a manipulation in the experiment that was not modeled: participants read a passage about the importance of cooperation before the start of the game. Our revised model is able to explain Lount et al.'s findings based on the dynamics of state trust. Reestablishing trust after a breach is a long process. In the case of early breaches, most of the rounds of the interaction are used to (slowly) reestablish trust. In the case of late breaches, most of the trust accumulates before the breach, leaving a smaller number of rounds of interaction to be damaged by the breach. This is consistent with results from the impression formation literature, emphasizing the importance of making a good first impression (Ambady & Rosenthal 1993).

Black-hat/white-hat effect

De Melo, Carnevale, and Gratch (2011) had participants play Prisoner's Dilemma with two different computerized confederate agents (cooperative & individual). Each agent was represented by a different animated face. Both agents used the same strategy (Tit-for-Tat), but displayed different facial expressions, representing different emotional reactions, to particular outcomes during the game (e.g., the cooperative agent expressed joy after instances of mutual cooperation and the individual agent expressed joy after instances unilateral defection). The authors suggested that participants used reverse appraisal to identify, from the agents' emotional displays, what the intentions of the agent were. The cooperative agent expressed emotions congruent with attempting to maximize the joint payoff of both players, whereas the individual agent expressed emotions congruent with attempting to maximize its own payoff. Participants played 25 rounds with each of the confederate agents in one of two orders, the cooperative agent then the individual agent (C-I), or the individual agent and then the cooperative agent (I-C). Given that the strategy of the two agents was identical, trustworthiness could only be inferred from facial expressions. Other authors have also shown that the pattern of trust learning can be influenced by incidental learning from facial expression, eye gaze, etc. (e.g., Strachan, Kirkham, Manssuer, & Tipper, 2016). De Melo et al. (2011) found that participants were sensitive to the emotions displayed by the two agents: they cooperated more with the cooperative agent than with the individual one. In addition, they found evidence for the black-hat/white-hat effect, as defined in the previous section. We did not explicitly model the process of inferring trustworthiness from facial expressions. Instead, we added 12 parameters that translated particular emotions into specific amounts of evidence of trustworthiness and trust necessity. However, these parameters by themselves did not make the model exhibit the black-hat/white-hat effect. The key difference was made by the trait trust deviation parameter (TTD in Equation 2), which allowed the model to fit the human data (r = .86, *RMSD* = .11) and reproduce the black-hat/white-hat effect.

Conclusion and Future Work

We presented a cognitive model of learned trust that integrates several seemingly unrelated categories of findings from the literature and thus makes headway toward a unified theory of learned trust. The model cumulates learning from its history of interactions with multiple other models (trait trust), learning from its current interaction (state trust), and (sometimes) incidental learning from facial expressions. The model predicts that trust decays toward distrust in the absence of evidence of trustworthiness or untrustworthiness. Our future empirical work will aim to test this novel model prediction. Our future modeling work will focus on better specifying the relationship between the dynamics of trait trust in past interactions and the perception of trustworthiness in the current interaction.

Acknowledgments

This work was supported by The Air Force Office of Scientific Research grant number FA9550-14-1-0206 to IJ and Oak Ridge Institute for Science and Education to MC.

References

Ambady, N., & Rosenthal, R. (1993). Half a minute: Predicting teacher evaluations from thin slices of behavior and physical attractiveness. *Journal of Personality and Social Psychology*, 64, 431-441.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Berg, J., Dickhaut, J., & McCabe, K. (1995). Trust, reciprocity, and social history. *Games and Economic Behavior*, 10(1), 122-142.

Biele, G., Rieskamp, J., & Gonzalez, R. (2009). Computational models for the combination of advice and individual learning. *Cognitive Science*, *33*(2), 206–242.

Collins, M.G., Juvina, I., & Gluck, K.A. (2016). Cognitive Model of Trust Dynamics Predicts Human Behavior within and between Two Games of Strategic Interaction with Computerized Confederate Agents. *Front. Psychol*. 7:49.

Collins, M.G., Juvina, I., Douglas, G., & Gluck, K.A. (2015). *Predicting Trust Dynamics and Transfer of Learning in Games of Strategic Interaction as a Function of a Player's Strategy and Level of Trustworthiness*. Paper presented at Behavior Representation in Modeling and Simulation (BRiMS) conference.

De Melo, C.M., Carnevale, P, & Gratch, J. (2011). The Impact of Emotion Displays in Embodied Agents on Emergence of Cooperation with People. *Presence*, *20*(5), 449–465.

Dirks, K.T., & Ferrin, D.L. (2001). The role of trust in organizational settings. *Organizational Science, 12*, 450-467.

Gonzalez, C., Lerch, F. J., & Lebiere, C. (2003). Instance-based learning in real-time dynamic decision making. *Cognitive Science* 27 (4), 591-635.

Harris, J. (2008). Maximizing the utility of MindModeling@ Home resources. Presented at Integrated Design and Process Technology Conference.

Harris, P.L., and Corriveau, K. (2011) Young children's selective trust in informants. *Phil. Trans. R. Soc. B*, 366, 1179–1187.

Helson, H. (1964). *Adaptation-level theory*. New York: Harper & Row.

Hilty, J., & Carnevale, P. (1993). Black-hat/white-hat strategy in bilateral negotiation. Organizational Behavior and Human Decision Processes, 55(3), 444–469.

Hoff, K. A., & Bashir, M. (2015). Trust in automation: Integrating empirical evidence on factors that influence trust. *Human Factors 57*(3), 407-434.

Hommel, B. & Colzato, L.S. (2015) Interpersonal trust: an event-based account. *Front. Psychol*. 6:1399.

Juvina, I., Lebiere, C., & Gonzalez, C. (2015). Modeling trust dynamics in strategic interaction. *Journal of applied research in memory and cognition. 4*(3): 197-211.

Juvina, I., Saleem, M., Martin, J. M., Gonzalez, C., and Lebiere, C. (2013). Reciprocal trust mediates deep transfer of learning between games of strategic interaction. *Organ. Behav. Hum. Decis. Process.* 120, 206–215.

Lee, J. D., See, K. A. (2004). Trust in automation: Designing for Appropriate Reliance. *Human Factors* 46(1): 50-80.

Lewicki, R. J., McAllister, D. J., & Bies, R. J. (1998). Trust and distrust: New relationships and realities. *Academy of management Review*, 23(3), 438-458.

Lount, R. B., Zhong, C. B., Sivanathan, N., & Murnighan, J. K. (2008). Getting off on the wrong foot: The timing of a breach and the restoration of trust. *Personality and Social Psychology Bulletin*, *34*(12), 1601-1612.

Lyons, J.B., Stokes, C.K., & Schneider, T.R. (2011). Predictors and outcomes of trust in teams. In Stanton, N.A. (Ed.) *Trust in military teams*. Ashgate Publishing Ltd.

McKnight, D.H., Cummings, L.L., & Chervany, N.L. (1998). Initial trust formation in new organizational relationships. *The Academy of Management Review* 23(3), 473-490.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J.R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 1-55). Hillsdale, NJ: Erlbaum.

Rapoport, A., Guyer, M. J., & Gordon, D. G. (1976). *The 2×2 game*. Ann Arbor, MI: The University of Michigan Press.

Rousseau, D. M., Sitkin, S. B., Burt, R. S., & Camerer, C. (1998). Not so different after all: A cross-discipline view of trust. *Academy of management review*, *23*(3), 393-404.

Schaefer, K.E., Chen, J.Y.C., Szalma, J.L., & Hancock, P.A. (2016). A Meta-Analysis of Factors Influencing the Development of Trust in Automation. *Human Factors*.

Sitkin, S. B., & Roth, N. L. (1993). Explaining the limited effectiveness of legalistic "remedies" for trust/distrust. *Organization science*, 4(3), 367-392.

Skowronski, J. J., & Carlston, D. E. (1989). Negativity and extremity biases in impression formation: A review of explanations. *Psychological Bulletin*, **105,** 131–142.

Slovic, P. (1993). Perceived risk, trust, and democracy: A systems perspective. *Risk Analysis*, **13,** 675–682.

Strachan, J., Kirkham, A., Manssuer, L., & Tipper, S. P. (2016). Incidental learning of trust: Examining the role of emotion and visuomotor fluency. *Journal of Experimental Psychology: Learning, Memory & Cognition*.

Sturgis, P., Read, S., & Allum, N. (2010). Does intelligence foster generalized trust? An empirical test using the UK birth cohort studies. *Intelligence*, 38(1), 45-54.

Yamagishi, T., Kikuchi, M., & Kosugi, M. (1999). Trust, gullibility, and social intelligence. *Asian Journal of Social Psychology*, 2(1), 145-161.

Yaniv, I., & Kleinberger, E. (2000). Advice taking in decision making: Egocentric discounting and reputation formation. *Organizational Behavior and Human Decision Processes*, *83*(2), 260–281.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

A minimal model of eye movement applied to visual search and change detection

Shane T. Mueller(shanem@mtu.edu) **Yin Yin Tan (yinyint@mtu.edu)**

Hannah North (hbossele@mtu.edu) **Kelly Steelman (steelman@mtu.edu)**

Department of Cognitive and Learning Sciences
Michigan Technological University
Houghton, MI 49931 USA

Abstract

We describe the Eye Movement Minimal Model-Modified (EM4), a lightweight minimally-sufficient model of eye movements that accounts for visual search times in several distinct paradigms. The model allows visual search to be guided by probe-item similarity in different foveal zones, which enables the model to be used as a front-end for various models of visual saliency. We apply the model to four distinct paradigms to demonstrate its flexibility and utility.

Keywords: eye movements; visual search; change detection

Background

In recent years, detailed models of visual processing that represent or are inspired by the human visual system have proliferated, providing many alternate computational approaches to investigating properties of visual attention, saliency, image analysis, and the like (Bruce & Tsotsos, 2009; Itti, Koch, & Niebur, 1998; Wolfe, Cave, & Franzel, 1989). Currently, more than 60 distinct methods of evaluating visual saliency have been compared using on popular benchmark (Bylinskii et al., 2016). In contrast, relatively less attention has been paid to modeling the mechanisms and strategies involved in directing visual attention via eye movements to perform visual search. Although some models of visual saliency have included foveated eye movements (e.g. Itti et al., 1998), models of visual saliency that ignore foveation and eye movement may make either unnecessary or unrealistic assumptions. Hornoff & Halverson (2003; 2004a, 2004b, 2007, 2011), developed and enhanced models of visual search via foveated eye movements using the EPIC computational architecture (Kieras & Meyer, 1997). As part of this effort, they described a "Minimal Model" involving the assumptions they felt most necessary and sufficient for modeling visual search in applied settings. Subsequently, more advances have been made to these models, both at the architectural and strategic level (Kieras, 2011; Kieras, Hornof, & Zhang, 2015), and these developments have been mirrored by a series of models using the ACT-R architecture (e.g. Salvucci, 2001; Nyamsuren & Taatgen, 2013; Choi, Han, Oh, & Myung, 2015). Yet the minimal model is an attractive target for practical simulation modeling outside the context of a cognitive architecture. Its notions have been adopted by several applied models of visual attention (e.g. Teo & John, 2008), but the model was not designed to handle visual search based on saliency and similarity cues, and so its lessons have not been widely as adopted in the broader field of computational vision that has otherwise led to dozens of visual and image-processing models that identify saliency.

In this paper, we describe the Eye Movement Minimal Model-Modified (EM4), which takes the Halverson & Hornoff model as a starting point, implements it as a standalone simulation model. To handle search in more general situations, the model incorporates search based on probe-item similarity, a quantity akin to what many visual saliency models produce naturally as an output, either via an activation or posterior probability distribution. After describing the model, we will show its ability to capture data in several related visual search and flicker change detection paradigms, illustrating its flexibility and utility.

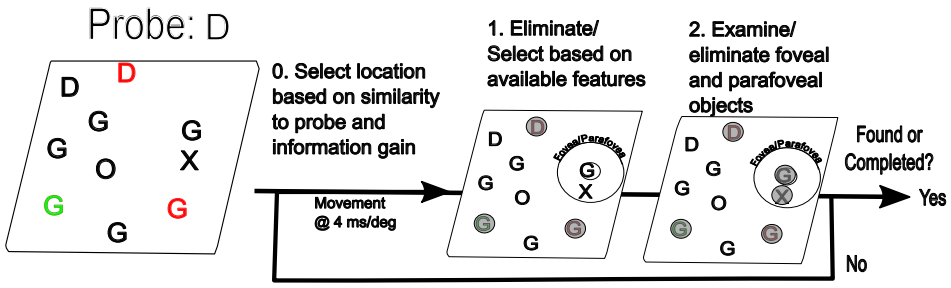
The Eye Movement Minimal Model-Modified

The EM4 is intended as a simple implementation and extension of the core assumptions of the "Minimal Model" proposed by (Halverson & Hornof, 2007), with the goal of accounting for major phenomena in visual search paradigms. The EM4 is implemented as a standalone software routine in the statistical computing language R, so that it can be repurposed and adapted to work with other models of visual processing or human performance, and serve as a lightweight modeling and teaching tool. The source code for the model is available via `https://github.com/stmueller/em4`. The basic stages of the model are shown in Figure . The model operates by simulating the timing of a series of eye movements and other decisions that produce a response in the task.

Primary assumptions

We assume that visual search involves a repeated set of stages in which a target object is selected based on its similarity to the probe and its potential for information gain, following which objects are eliminated or selected based on their similarity to the probe. This repeats (fixation target is selected and foveated, items are eliminated or selected based on similarity) until either the probe object is found or a decision is made to stop search. These probe-item similarity values are *inputs* to the model and we treat them as free parameters. This is a departure from the original description of the minimal model, which used identity-match, and subsequent EPIC visual search models, which have also used feature-level descriptions to represent how different types of information are available at difference eccentricities. Use of similarity provides a useful mid-level representation, such that low-level feature-based visual processing models could produce similarity as an output, perhaps without even requiring those models to be directly embedded within the simulation. The basic

Figure 1: Schematic stages of model. Once fixated (0), target locations are eliminated and selected (1) based on similarity-to-probe. Here, color mismatches may be quickly eliminated in the periphery. Next, (2) targets in the fovea and parafovea are examined, and neighboring targets not eliminated are examined. Once no more targets appear in the parafovea, a new location is selected based on available similarity-to-probe and information gain. The process is repeated until search is complete.



assumptions of the model include:

Information is represented as probe-to-item similarity. The main paradigms investigated by Halverson & Hornoff involved locating a text-based menu item that was always present. Yet even Latin characters have a well-described similarity space (Mueller & Weidemann, 2012), such that similar characters are confusable and take longer to discriminate. In more traditional visual search tasks, targets that are distinguished by a single feature can produce visual 'pop-out', such that the number of distractors does not impact search time. Furthermore, some items in the periphery might be selected or ignored based on similarity to the probe–if the search target is an "O", any "X" in the periphery might be ignored, but a "U" might require more investigation. Consequently, the EM4 represents this information as a similarity score, whose values may differ foveally, parafoveally, and peripherally.

Zones of detection. In reality, the availability of color, shape, size, and location of objects degrade differentially and smoothly as an object's eccentricity increases, with serious degradation starting to occur 30°–45°from the fovea (Boff & Lincoln, 1988). Many recent models have used an eccentricity function (Kieras, 2010; Nyamsuren & Taatgen, 2013), which parametrically defines the availability of different features at different eccentricities, but the EM4 retains just three visual zones: the fovea with radius 1°; the parafovea with typical radius 3.5°; and the periphery which involves the remaining visual field. We assume that the location of visual objects is available everywhere, insofar as any object can be selected as an eye movement destination. Foveated targets can be identified explicitly (with some chance of error), whereas parafoveal targets with high probe-item similarity are more likely than those with low similarity to be selected for subsequent eye movements. In practice, because relevant objects can be detected or rejected parafoveally, the size of the parafovea maps roughly onto the useful field of view (UFOV; Edwards et al., 2006), and the size may depend on properties of the task. In addition, just as peripheral objects that are high in probe-item similarity might direct subsequent eye movements to that location, those high in probe-item *dissimilarity* can be used to eliminate targets from consideration and thus make fast rejection responses (see Chun & Wolfe, 1996). In the fovea, the similarity represents the *probability* that a particular object is identified as the target. For true targets, this maps onto the probability of misdetection used by Halverson & Hornoff, but also permits false alarms if the value is nonzero for foils.

Movement and decision times. We assume that after each fixation, a decision is made about whether the searched-for target has been identified, following which a choice is made about the next eye movement destination. The timing of this decision-action cycle constitutes one of the main free parameters of the model, which we assume is impacted by the nature of the stimuli, as well as dynamic aspects of the environment. Although the original EPIC models attempted to use fixed architectural parameters to determine this timing, different data sets require using some very different decision timing. In addition, this time incorporates all time that is constant with each foveation. A saccadic eye movement is assumed to take place at 4 ms/degree of visual angle.

Accepting and rejecting matches. In general, the existence of visual pop-out is taken as evidence that some decisions can be made based on information in the visual periphery. In the tasks described here, detection of high-similarity targets outside of the fovea lead to a subsequent eye movement to the target to confirm and localize the target (partly because most of the tasks we examine require a selection of the target via mouse movement). However, just as a target can be identified in the periphery or parafovea, we also assume that targets can be rejected from consideration based on information in the periphery or parafovea. In the parafovea and periphery, the similarity score represents an activation level, such that values above 0.5 represent greater similarity to a probe, indicating a possible match; values below 0.5 indicate dissimilarity to a probe great enough eliminate from search. Targets with periphery similarity greater than 0.5 each need to be examined and either eliminated or responded to if found to be identical to the target. If all peripheral targets with similarity above 0.5 are examined and eliminated, a neg-

ative response can be made. However,for peripheral targets, we assume that target-probe values below 0.5 permit eliminating the target without eye movement (i.e., preattentively), allowing for fast responses. This accounts for findings such as the ability to make a probe-absent decision without examining each target, or (when a probe should produce pop-out) to make a target-absent decision quickly even when nothing is detected (Chun & Wolfe, 1996). Importantly, only probe-item similarity is used directly, and the model is not impacted by target-distractor similarity, which may provide additional gestalt cues for helping to identify and classify oddball search targets or possibly make search less efficient.

Selecting subsequent locations for search. Deciding where to search next (including in cognitive search of memory, physical search of environments, and other domains) involves cost-benefit analysis (Perelman & Mueller, 2015), because the costs of moving must be weighed against the potential gain in information (Drury, 1975; Bruce & Tsotsos, 2009). For search constrained by eye movements, the time needed to move the eye to a new destination is relatively insensitive to the distance moved (only 4 ms/degree), in contrast to the fixed cost of 100 ms or more required to program and execute the movement, and the time required to classify an item once it is foveated. However, deliberate short eye movements help avoid repeated search of a location by making the task of keeping track simpler; this may improve time-to-find, even if a more distant location could offer maximum gain in information, so that a new location with more potential targets may be better than a closer location with only one target.

The present model balances these by first looking for high-similarity unidentified targets in the parafovea; if this fails, it computes a neighborhood activation score for each unvisited target (the sum of the exponentially-discounted similarity of all unvisited nearby targets), and deciding the next eye movement based on a mixture of the normalized inverse neighborhood similarity scores and noisy distance-to-target, so that the next target may (at one extreme) be the next-most-similar, or (at the other extreme), be based purely on the distance selection scheme proposed by Hornoff & Halverson. When target decisions are made based on discounted neighborhood activation, this favors movements to targets in dense regions where a single fixation is able to eliminate several objects.

In summary, the model implements a stand-alone version of the minimal model that operates by repeatedly selecting objects, fixating on them, and eliminating them from contention, until the selected target is found or all targets are eliminated. Next, we will examine how the model fits several related visual search paradigms.

Model Fits to Data

In this section, we will describe the model's fit to several empirical data sets. These include a menu search task, a visual search task, and two flicker-based change detection tasks. The parameter values and goodness-of-fit values (both R and percent deviation, where appropriate) are shown in Table 1. We will examine a menu search task, a feature-search task; and two flicker-change blindness tasks whose performance profiles differ substantially.

Figure 2: Tasks modeled in this paper. A. Menu selection task; B. feature search; C. Dot-flicker change detection; D. Sparse change detection.

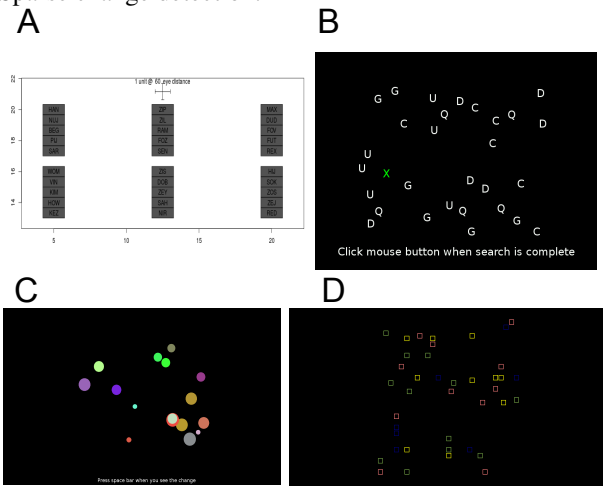
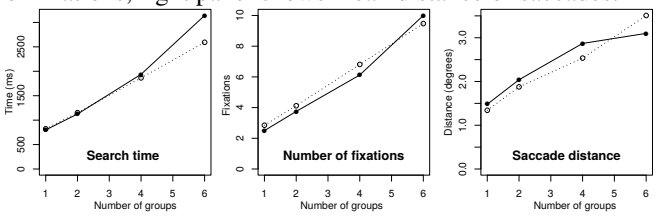


Figure 3: Model and data from menu search task. Left panel shows search time (in ms); center panel shows mean number of fixations; right panel shows mean distance of saccades.



Menu Search Task

The primary task used by Halverson & Hornoff to inform recommendations for a minimal model involved menu search, in which blocks of contiguous text-labeled targets needed to be searched to find a specific target (see Figure 1a). In this task, aside from target location, no information in the parafovea or periphery is useful for localizing the target menu, as the labels were 3-letter strings that could not be easily identified without foveating on or near the menu.

Method This task involved search conditions involving 1, 2, 4, or 6 blocks of menu items, where each block consisted of five items spaced vertically at .66°, arranged in up to three columns, two blocks per column, with a vertical separation of 1.33°and horizontal separation of 7.5°between blocks. Each target had a unique 3-letter label, and on each trial, a participant searched for a specific labeled target. Three critical dependent measures were examined: mean time to find, mean number of fixations, and mean saccade distance. Full parameters are shown in Table 1. Model fits are shown in Figure 3.

Discussion This simulation produced good fits, using parameters and assumptions similar to Halverson and Hornof (2007), the one major exception is the noise parameter used to select subsequent target locations, which was much larger for the present model, primarily because in the present model, eye movements are first made to nearby locations in the parafovea, rather than solely on a noisy-distance scheme. This gives the model a natural preference for nearby objects, and so to counteract this, a larger default noise parameter was required. This illustrates that the EM4 captures the major phonemonon or which the original minimal model was designed to account for.

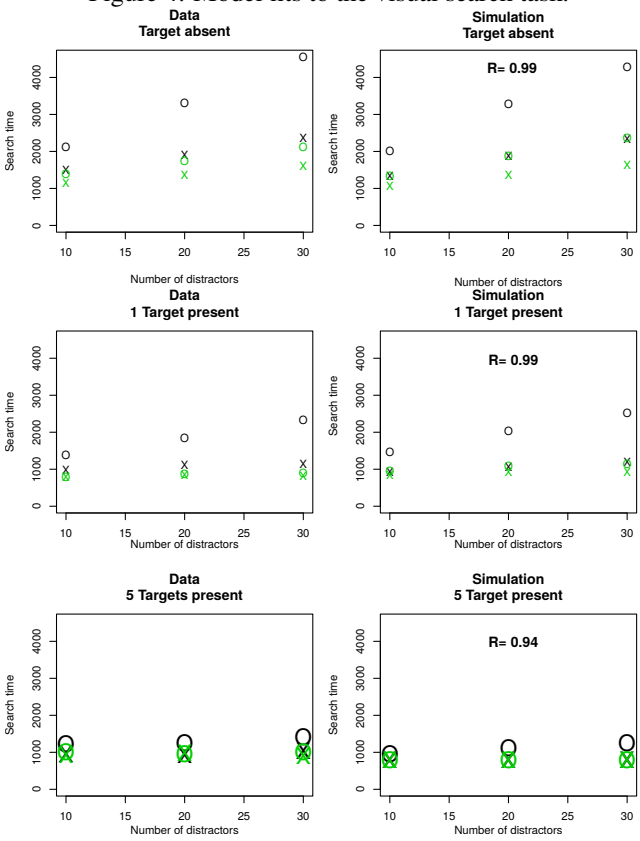
Visual search of simple targets with pop-out

Although the previous task is a useful starting point, it differs from the most commonly-used visual search paradigms typically used within vision science and psychology. Such search tasks typically differ in three ways from this menu search: (1) they involve haphazard stimulus arrangement, making systematic search more difficult; (2) they often involve search for a specific target character amongst a field of distractors that may be either similar or dissimilar to the target (i.e., searching for a T in a field of Ls or Os); and (3) they are often used to demonstrate visual pop-out or feature search, the finding that the presence or absence of some features can be detected peripherally. Thus, the next step in developing the model was to examine how it can account for a more traditional visual search task.

Method This study involves an implementation of a visual search task with several search targets producing visual popout (Mueller & Piper, 2014, see Figure 1b), in a cross-national study (Tan, 2016) that involved 136 participants. In this task, participants searched for a specified target on each trial (a white or green O or X) in a field of either 10, 20, or 30 round white characters (C, D, G, Q, and U) on a black background that was approximately 15°x 10°of visual angle. On different trials, 0, 1, or five targets were present. These conditions parametrically varied the efficiency of search, the number of distractors, and the number of targets, and provided a systematic data set for modeling search times. On each trial, the field of elements was presented until the participant clicked the mouse button; after which the elements were each replaced by a circle, and the participant was instructed to either indicate the location of an object matching the probe, or a label marked "none" if no objects matched the probe.

Results Accuracy for the task was high (98.5%) and so we will consider only search times, which are shown in Figure 4. The human results (left column) show that response time for rejecting pop-out targets tended to become longer with larger search sets, with a clear ordering from most difficult to least of white O, white X, green O, and green X. The same ordering occurred regardless of whether a target was present or absent, but the times were faster (and the slope with respect to number of distractors was smaller) when a target was present, and this diminished further when multiple targets were present.

Figure 4: Model fits to the visual search task.



The fits to data were good (see Table 1), with several differences in parameter values from the first model being (1) smaller detection/rejection time parameters were used, and (2) eye movements locations were selected based on probe-item similarity, rather than by distance alone; and (3) failure-to-detect was reduced to 0.0. The smaller detection times are reasonable because the current task required detecting a single letter instead of a 3-letter sequence. The use of probe-item similarity (or a similar concept) is necessary to fit these data, and this required making assumptions about probe-item similarity for each target class (green and white Xs, Os, and D/G/U/Q/C) in each zone. The slope of the response time to each target class (with respect to number of distractors) is primarily controlled by the peripheral probe-item similarity. These similarity values were assigned with a uniform distribution having a range of 0.3 units, with the minimum values of .45 (when color and shape match), .3 (when either color or shape match) and .25 (when color and shape mismatch). Thus, the small but positive slope for target-absent responses arises because as the number of distractors increase, the number of distractors with a probe-target similarity above 0.5 increases, requiring additional eye movements to eliminate. Together, these assumptions accounts for search times with a mean proportional absolute deviation of 0.16 and a correlation of .94.

Discussion The visual search data shows that the EM4 can provide a credible account of a more traditional search task, including pop-out effects, target-absent effects, and effects of the number of distractors. Thus, the model is capable of predictions in two major visual search paradigms, including one in which involves parallel feature-based selection and elimination of targets. The model predicts timing of the search task well, but without recording eye movements, there is often a potential for trading off zone size (fovea and parafovea) with dwell time. For example, if the parafovea were twice as large but the dwell time was doubled, a similar fit might be obtained. Along with measuring eye movements directly, another way to constrain the model is with a flicker-based search task, a commonly-used search paradigm that yokes eye movement times to a fixed frequency of presentation. We will next examine two flicker tasks to demonstrate the model's flexibility and help constrain its assumptions about timing of movement. On their surface, the two tasks appear very similar, but produce performance profiles that differ substantially. Thus, it will be important to examine the aspects of the model that change in order to account for these across-task differences.

Flicker-paradigm change detectione

A commonly used paradigm involving visual search is the flicker change detection task (see Rensink, O'Regan, & Clark, 1997), in which two visual stimuli (either artificial or natural) that differ in some small way are shown repeatedly in succession, with a brief empty 'flash' between them (e.g., 50 ms) that prevents low-level visual change detection and requires deliberate search among the targets to find the difference. Here, even if detection can be done quickly, participants often cannot benefit from more than one eye movement per flash, which constrains the rate of information search. In this experiment, we examined a relatively difficult version of the task that incorporated four different types of change while varying the number of distractors.

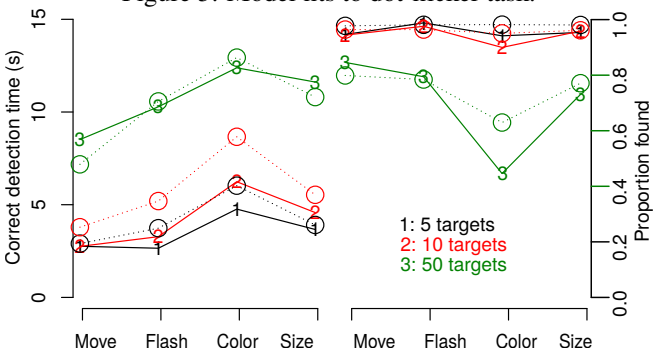
Methods. The present data was collected on the same groups of participants as the visual search task. This task used a modified version of the PEBL dot-flicker change blindness task (changeblindness), and involved three stimulus conditions with 5, 10, or 50 distractors. Each trial involved one of four change types: a color change, a size change, a position change, or a target disappearance. The entire stimulus visual field was approximately 20°x 15°. Each display frame appeared for approximately 450 ms, followed by a 50 ms blank flash, which was sufficient to disrupt low-level visual cues of change. Participants were permitted a maximum of 30s to find the change, which they then indicated by clicking with the mouse at the location of change.

Modeling the task differs from the previous models in that there is no known a priori probe, so the notion probe-item similarity is not applicable. Consequently, for the model, we interpret the zone similarity values to indicate the available evidence for a change across the flicker mask. The model assumes that no real information is available in the periphery, but evidence for a flicker-change will often be detectable in the parafovea (targets have similarity around .8 whereas non-targets have similarity around .6), which can then direct a foveation to confirm and localize the change. Because the rate of search is constrained, the main dependent timing measures are constrained by the effective size of the parafovea, which we adjusted to improve fit to data.

Results. Although small differences were observed in time and accuracy between 5 and 10-target displays, participants were considerably less accurate and slower on the 50-target display. As shown in Figure 5, the flash condition was slower and less accurate than the move condition (a move is essentially a double-flash), and size-change tended to be about as difficult as the flash condition. Color change was by far the most difficult condition. The model produced reasonable fits to the data, although it overpredicted the time needed to find the target on the smaller displays. The model assumes that the difficulty of different conditions arises because of a failure to detect changes of different types once a potential change is foveated, as shown in Table 1. In addition, the parafovea zone had a radius of 5.5°. This indicates that such changes may be available quite far from the fovea, but may often go undetected, which would require frequent revisits to previous locations.

Before discussing the results of the change detection task, we will examine a second study using an alternate version of the task that employs top-down cueing, and thus permits probe-item similarity to play a role in the search task.

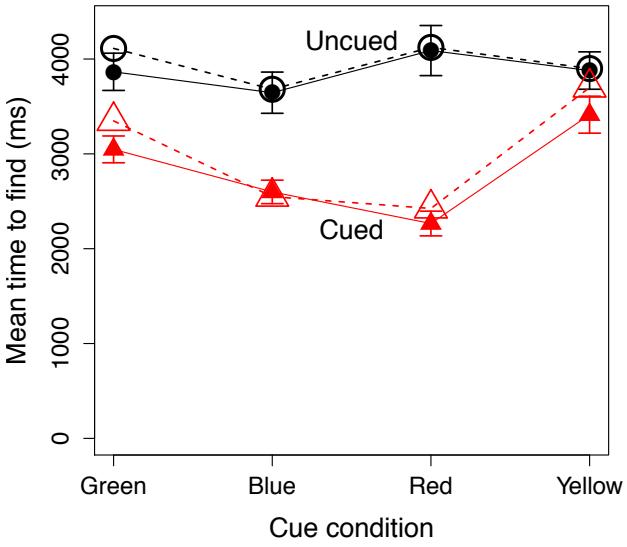
Figure 5: Model fits to dot-flicker task.



Top-down control in flicker-based change detection

Methods The final data set also used the flicker paradigm, but differed from the previous task in several ways. First, the field of view was larger (28°x 28°), and only one type of change occurred (a single target appeared and disappeared). On each trial 40 symbols appeared, drawn randomly from a set of four colored symbols (red, green, blue, and yellow squares). The larger field of view and more uniform targets made the task substantially easier, perhaps because of crowding and spacing effects (see Pelli, 2008). On half the trials, a color cue was given indicating the color of the change.

Figure 6: Model fits accounting for top-down control in change detection task



Results The single change type, coupled with the more dispersed display, produced a much easier task than the previous one: mean time to find in the 40-target task was under 4 s, comparable to the 10-target condition in the previous experiment. Furthermore, accuracy was close to 100%; in contrast to the 50-80% accuracy produced in the 50-target condition of the previous study. In addition, when cued, the time was reduced further, although the advantage for different colors differed, depending on the color.

To model these data, we assumed that different colors produce a different probability of detecting a change in the fovea, similar to the previous model. However, on cued trials, probe-item similarity is used to eliminate potential targets and constrain search. The best case scenario reduces target locations to around 10, but because of random sampling and layout of points, this does not reduce the time-to-find to 1/4 of the original. The model accurately predicts the that scale of this reduction is about 1/3. Differences in the color cue conditions were modeled by adjusting the peripheral similarity of different colors to the cue, so that in the case of red, typically 10/40 targets needed to be searched, but in the case of yellow and green, closer to 20/40 targets needed to be searched (because of their similarity). In these models, yellow and green often cannot be distinguished rapidly in the periphery, and so more of these targets were foveated to eliminate the foils. Overall, although some of the detection parameters were substantially different from the previous experiment, the model produces reasonable fits for both with interpretable changes in parameters (see Table 1). As in the previous change blindness model, parafovea size was slightly larger than the search task–in this case 4.0°. This is a consequence of the fact that search times on the order of 3-4 s necessarily involve at most 6 to 8 foveations, and the only way to reliably cover the visual field is if each foveation obtains information from this area. Thus, both change detection models suggest that change can be detected at 4 or more degrees from fixation, and that color-changes can be especially easy to miss (with failure rates af around 40% when fixated).

Parameters of models

Table 1 summarizes the main parameters used across tasks. Results are mainly impacted by assumptions about how long each detection/decision phase take, and the probability of detecting information in different visual zones-especially probability of detection failure in the fovea.

Discussion

The EM4 adopts and adapts the minimal model assumptions proposed by Halverson & Hornoff (2002), and extends the model to capture primary effects of several visual search paradigms. We have demonstrated the effectiveness of the model against four data sets, and the parameter settings for these models provide insight into the timing, accuracy, and availability of information. We will conclude by identifying some of the main lessons we have learned from these models.

Lessons of the models

Fovea zones. Accurate prediction of times and accuracies in search tasks require accounting for the information available in different foveal zones, and decisions about both presence and absence of this information.

Peripheral information. Substantial information about both presence and absence of information is available in the visual periphery (i.e., for pop-out tasks) and parafovea (for all tasks), and search is often guided by presence and absence of this information in all three zones.

Target rejection. Rejection of targets frequently occurs without foveation; identification of targets often is coupled with foveation.

Parafoveal preference. Search times can typically be adequately accounted for by a model that attempts to first confirm any high-likelihood targets parafoveally, and then maximize information gained in each subsequent movement.

Probe-item similarity. Probe-item similarity is useful in predicting a number of effects of visual search, sot that models of visual salience may benefit from incorporating probe information.

Conclusions. The EM4 intends to be a simple minimalistic model of foveated eye movement search. It is a standalone model, and so it may be useful for lightweight practical evaluation in human factors domains, as a simulation model education contexts, and as a lightweight front end for visual processing models that produce activation or posterior probability scores that can be interpreted as a probe-item similarity.

References

Boff, K. R., & Lincoln, J. E. (1988). *Engineering data compendium. human perception and performance* (Tech.

Table 1: Distinct parameters settings used in different paradigms.

Model	Fixation Time	Response Time	Parafov. radius	Detection failure	Parafov. Similarity	Peripheral Similarity	Scaled Deviation	*R*
Menu search	250	100	3.5	.09	.7	.6	.08/.065/.091†	
Visual search	200	500	3.5	0	.7	.45,.3,.3,.25	.16	.94
Change Detection I	500	100	5.5	.05,.05,.4,.09	.775-.825	.6	.16/.063‡	.97
Change Detection II	500	100	4.0	.04,.02,.05,.04	.7/.85	.48,.46,.4,.5	.045	.98

Note: Scaled deviation is mean of absolute error divided by observed value; *R* is Pearson's correlation. Multiple values in each parameter cell indicate values for distinct conditions. †Values for response time, fixations, and saccade distance, respectively. ‡Values for response time and accuracy.

Rep.). Wright-Patterson Air Force Base, OH: Armstrong Aerospace Medical Research Laboratory.

Bruce, N. D., & Tsotsos, J. K. (2009). Saliency, attention, and visual search: An information theoretic approach. *Journal of vision*, *9*, 1–24.

Bylinskii, Z., Judd, T., Borji, A., Itti, L., Durand, F., Oliva, A., et al. (2016). *MIT saliency benchmark.* http://saliency.mit.edu/.

Choi, Y., Han, J., Oh, H., & Myung, R. (2015). Cognitive model of human visual search with saliency and scene context for real-world images. In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 59, pp. 706–710).

Chun, M. M., & Wolfe, J. M. (1996). Just say no: How are visual searches terminated when there is no target present? *Cognitive psychology*, *30*(1), 39–78.

Drury, C. G. (1975). Inspection of sheet materialsmodel and data. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *17*(3), 257–265.

Edwards, J. D., Ross, L. A., Wadley, V. G., Clay, O. J., Crowe, M., Roenker, D. L., et al. (2006). The useful field of view test: normative data for older adults. *Archives of Clinical Neuropsychology*, *21*(4), 275–286.

Halverson, T., & Hornof, A. J. (2004a). Local density guides visual search: Sparse groups are first and faster. In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 48, pp. 1860–1864).

Halverson, T., & Hornof, A. J. (2004b). Strategy shifts in mixed-density search. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society* (pp. 529–534).

Halverson, T., & Hornof, A. J. (2007). A minimal model for predicting visual search in human-computer interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 431–434). ACM.

Halverson, T., & Hornof, A. J. (2011). A computational model of "active vision" for visual search in humancomputer interaction. *Human-Computer Interaction*, *26*(4), 285–314.

Hornof, A. J., & Halverson, T. (2003). Cognitive strategies and eye movements for searching hierarchical computer displays. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 249–256). ACM.

Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. on Pattern Analysis & Mach. Intelligence*(11), 1254–1259.

Kieras, D. E. (2010). Modeling visual search of displays of many objects: The role of differential acuity and fixation memory. In *Proceedings of the 10th international conference on cognitive modeling* (pp. 127–132).

Kieras, D. E. (2011). The persistent visual store as the locus of fixation memory in visual search tasks. *Cognitive Systems Research*, *12*(2), 102–112.

Kieras, D. E., Hornof, A. J., & Zhang, Y. (2015). Visual search of displays of many objects: Modeling detailed eye movement effects with improved epic. In *Proceedings of the 13th international conference on cognitive modeling.*

Kieras, D. E., & Meyer, D. E. (1997). An overview of the epic architecture for cognition and performance with application to human-computer interaction. *Human-computer interaction*, *12*(4), 391–438.

Mueller, S. T., & Piper, B. J. (2014). The psychology experiment building language (PEBL) and PEBL test battery. *Journal of neuroscience methods*, *222*, 250–259.

Mueller, S. T., & Weidemann, C. T. (2012). Alphabetic letter identification: Effects of perceivability, similarity, and bias. *Acta Psychologica*, *139*(1), 19–37.

Nyamsuren, E., & Taatgen, N. (2013). Pre-attentive and attentive vision module. *Cogn. Sys. Research*, *24*, 62–71.

Pelli, D. G. (2008). Crowding: A cortical constraint on object recognition. *Current opin. in neurobiology*, *18*, 445–451.

Perelman, B., & Mueller, S. T. (2015). Identifying mental models of search in a simulated flight task using a pathmapping approach. In *Proc. of the 18th International Symposium on aviation psychology (ISAP 2015)* (pp. 398–403).

Rensink, R. A., O'Regan, J. K., & Clark, J. J. (1997). To see or not to see: The need for attention to perceive changes in scenes. *Psychological science*, *8*(5), 368–373.

Salvucci, D. D. (2001). An integrated model of eye movements and visual encoding. *Cognitive Systems Research*, *1*(4), 201–220.

Teo, L., & John, B. E. (2008). Towards a tool for predicting goal-directed exploratory behavior. In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 52, pp. 950–954).

Wolfe, J. M., Cave, K. R., & Franzel, S. L. (1989). Guided search: an alternative to the feature integration model for visual search. *Journal of Experimental Psychology: Human perception and performance*, *15*(3), 419. (01755)

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Towards a General Model of Repeated App Usage

Sabine Prezenski (sabine.prezenski@tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems, Technische Universität Berlin
10587 Berlin, Germany

Nele Russwinkel (nele.russwinkel@tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems, Technische Universität Berlin
10587 Berlin, Germany

Abstract

The main challenge of implementing cognitive models for usability testing lies in reducing the modeling effort, while including all relevant cognitive mechanisms, such as learning and relearning, in the model. In this paper we introduce a general cognitive modeling approach with ACT-R for hierarchical, list-based smartphone apps. These apps support the task of selecting a target, via navigating through subtargets positioned on different layers. Mean target selection time for repeated app interaction, learning and relearning behavior was collected in four studies conducted with either a shopping app or a real-estate app. The predictions of the general modeling approach match the empirical data very well, both in terms of trends and absolute values. We also explain how such a general modeling approach can be followed. The presented general model approach requires little modeling effort to be used for predicting overall efficiency of other apps. It supports more complex interface, as well.

Keywords: ACT-R; usability; apps; cognitive modeling; learning; relearning; updates; general model

Introduction

Numbers of smartphone apps are growing and so is the need for efficient usability testing methods. Cognitive models simulate human behavior and can in theory be utilized either as a supplement to, or instead of real user testing. To achieve this aim, especially in terms of costs and effort, it is crucial to develop valid cognitive models for specific tasks and app characteristics. These models should be written in a general manner, in order to minimize the effort to transfer them to other similar apps. Such general models could then be used to predict specific usability measures like efficiency. This would be particularly helpful in the prototyping phase of apps, where these models could be used instead of user tests. Moreover, the model traces could provide evidence on potential (cognitive) causes of usability problems that are not achievable with user tests.

Theory

Smartphone apps often support a limited amount of tasks and although apps for a great variety of tasks exist, the structures and functionalities of these apps are similar. Consequently, predicting usability of such apps with a general cognitive modeling approach would be useful and worthwhile. In order to provide meaningful predictions, user behavior should be depicted accurately by such general cognitive models. It is crucial that these models are written in a manner that transferability to other similar, but not identical apps, in terms of content and structure, is feasible with minimal effort. Such an approach implicates that not all cognitive mechanisms of users are represented by the models. In respect to transferability, simplifications are necessary for such an approach.

ACT-R has a modularized structure, resembling the architecture of the human brain. Specified modules handle different types of information, called chunks. Each chunk has slots; this is where the smallest pieces of information are stored. The different modules interact via specialized buffers. Visual information is processed by the visual module and its two buffers (visual-object and visual-location). Motor movement is controlled by the manual module and its manual buffer. The declarative module serves as the systems memory and retrieved information from memory is stored in the systems retrieval buffer. The imaginal module and its correspondent buffer are required for learning new information. The steering of the model is governed by the goal module and buffer. The procedural module connects the modules and selects (production-) rules that steer the model behavior. A production is selected and executed, if the states of the buffers are met. The production then alters the states of the modules. Subsymbolic processes are also addressed in ACT-R. If a production requests a chunk and two chunks match the request, then the chunk with the higher activation level is selected. The activation level of a chunk depends on how long ago the chunk was created, on how often it was used and on when it was last accessed. Other parameters are the latency factor which influences the duration until a retrieved chunk is available in the retrieval buffer and the duration until a retrieval failure occurs. The later is also manipulated by the retrieval threshold parameter.

Box 1: A brief introduction to ACT-R

Hierarchical, list style apps are a common type of apps. They are often designed to support the task to find and select a target by navigating through different layers and selecting a subtarget on each layer. This paper presents a general cognitive model of a user interacting with hierarchical list style apps. The model covers repeated interaction, thus investigating learning and relearning effects.

Some cognitive modeling approaches addressing the usability of HMI already exist. The most prominent is CogTool (John, Prevas, Salvucci, & Koedinger, 2004). This is a rapid prototyping tool that enables the creation of cognitive models and predicts execution times for predefined task. But important aspects for the usability of apps such as version updates and learning behavior cannot be modeled with CogTool. A main objective of our work is to develop a model that learns through experience with the interface. A modeling procedure that is strong in

representing learning mechanism is the cognitive architecture ACT-R (for a brief outline of the mechanisms of ACT-R see box 1). Successful ACT-R models of menu and mobile interaction exist. The following aspects of these models are used our model. In an eye-tracking study using desktop computers, Byrne (2001) showed that menu search can be modeled with ACT-R. The fact that this model reads the menu from top to bottom is adopted in our model. In a study on learning of mobile phone usage for elderly novice users (Das & Stuerzlinger 2007), the performance increase in the model was due to the successful recall of locations of keys. Our model also uses the retrieval of locations as a learning mechanism. St.Amant, Horton, & Ritter (2007) developed a model that predicted time on task for expert users searching in hierarchical menus with a feature phone. Their assumptions that experienced users navigate with parent child chunks through hierarchies is implemented in a specialized chunk of our model.

The aim of this paper is to develop and test a general cognitive model with ACT-R that predicts user behavior during repeated interaction. Thus, the model incorporates learning and relearning mechanisms. The modeled task is repeated target selection with different hierarchical list apps.

Methods

Four studies were conducted with either a shopping app (*shopping 3-4, shopping 4-3)* or a real-estate app (*house apartment, apartment house*). Both apps are custom-designed android apps. The empirical studies are presented elsewhere in greater detail (Prezenski, Lindner, Moegele, & Russwinkel, in preparation.; Prezenski & Russwinkel, 2014). Since this paper focuses on the modeling approach, only a brief outline of the apps and the study procedure will be given. See figure 1 for an overview of the apps.

The main functionality of the shopping app is to compose a shopping list. To place products on the list, navigation through various stores and product categories in the menu is required. The real-estate app allows the selection of search criteria for real-estates, such as the number of rooms or the city district. Again, the criteria can be found by navigating through different categories. Both apps are multi-layer hierarchical list apps with variations in menu depth. The main functionality of the apps is target selection via navigating through a number of layers (see figure 1). On each layer a subtarget has to be preselected. For each target, there is only one correct path of subtargets leading to the target. Two versions of the shopping app are used: one with three and one with four layers of menu depth for all targets. As illustrated in figure 1, the path leading to the target e.g. *alcohol free beer* differs between the two versions. The real-estate app has a mixed number of layers (either three or four layers per target). Furthermore, the real-estate app is adaptive. Depending on preselection the paths leading to targets and the position of some subtargets changed. As can be seen in figure 1 the path leading to the target *lawn* differs if either house or apartment has previously been selected.

Figure 1: Screenshots of the apps with the modified paths leading to the targets for the different versions (shopping app) or different previous selections (real-estate app).

Task

Participants repeatedly selected targets using the apps installed on a Google Nexus 5 smartphone, running android 4.1.1. Targets were read to the participants and after selecting the target, participants were required to navigate back to the first layer of the app. In all four studies there were four runs, each run required the participants to select a number of targets. Participants of the studies *shopping 3-4*, and *shopping 4-3* had to select nine targets (products) per run. The same targets were used for all four runs. After the second run the version of the shopping app was updated, either a layer was added (*shopping 3-4*) or removed (*shopping 4-3*). Thus, the paths leading to the targets were the same for the first and second layer but were altered from the third layer on. Participants of the studies with the real-estate apps had to select six or seven targets (criteria) per run. Some of the targets were the same for all runs, e.g. numerical criteria such as *the rent* remained the same for all four runs. Others, like *the city district* varied between all four runs. Participants of the study *apartment house* searched for an apartment in the first two runs and then switched to searching for a house. The order was reverse for participants of the study *house apartment*. Due to the adaptive character of the real-estate app, the pre selection of house or apartment altered the position of the numerical criteria (e.g. the number of rooms) and also changed the path leading to lawn. This path differed for house and apartment from the second layer on.

Model

The data obtained with the studies *shopping 3-4* and *shopping 4-3* was utilized to develop the main model mechanisms and a first ACT-R model. The subsequent studies *house apartment* and *apartment house* were designed for two reasons: First, to test whether the model can predict data obtained with a different app and second, to ensure that the model mechanisms are held in a general matter. Thus, the model incorporates mechanisms for handling variations in depth within an app, changes in paths from varying layers on and variations in locations of targets and subtargets. The task of repeatedly selecting targets in multilayer applications is captured in the model.

Table 1: Examples of the chunk types of the model

meaning chunk NAME "SEARCH" OBJECT SEARCH	*association chunk* OBJECTS HOUSE CATEGORY SEARCH	*path chunk* FIRST SEARCH SECOND WHAT THIRD RENT FOURTH HOUSE TARGET-IM HOUSE COUNT FOUR
chunk with location SCREEN-POS VISUAL LOCATION35-0-0 VALUE "House" COLOR BLACK HEIGHT 10 WIDTH 28 TEXT T	*goal chunk* STATE PREPARECLICK SUBTARGET "SEARCH" FINALTARGET "HOUSE" TARGET-MEANING HOUSE MENUDEPTH FOUR IMMOLIST ("MOABIT" ....) MENUDEPTHLIST (THREE ...)	

Summary of Main Mechanisms[1]

Without prior experience with the specific target, visual attention is directed to the top of the page. For each visual processed word, a retrieval request for a *meaning chunk* containing the word as string and as meaning is made (see table 1 for examples of the chunk types used in the model). Navigation through the application is achieved via world knowledge, which consists of associations between two words (*association chunk*). For each read word the attempt to retrieve an *association chunk* with the target is made. If an *association chunk* containing the current word and the target is retrieved, a *path chunk is built*, holding the path leading to the target in the imaginal buffer and the word is selected. A *path chunk* consists of the slots *first, second, third* and *fourth* for the subtargets. The slot *target-im* holds the target word. The *count* slot of the *path chunk* holds information on the current menu depth and is changed if a different subtarget is required.

With experience with the specific target, navigating to this target is realized via the *path chunks* previously built. After a successful retrieval of a *path chunk* a chunk with the location of the relevant subtarget in the path is requested. The retrieved location is visually inspected and the subtarget is selected.

Model steering

Learning mechanisms are incorporated in the model. Furthermore, the model can handle a number of changes to the interface; such as version updates influencing all targets and smaller changes affecting only some targets *Model steering* is implemented with the goal in mind to reuse or extend the model for other applications. Thus, simplifications of some cognitive mechanisms and special chunk slots to account for interface variations are used. Model steering is realized via a *count* slot in the imaginal buffer, which holds the current depth and via different slots in the goal buffer. The *menudepth* slot holds information on changes in depth for the current target (e.g. number of layers leading to the target). The model can handle varying and constant depth values Currently, mechanism exist for a constant depth of three and two layers and for depth changing from three to four and vice versa, with the path leading to the target altered either from the second or from the third layer on. The *menudepth* slot is used to differentiate between strategies for the last versus the other layers in the path leading to the target. The *menudepth* slot is also required after an error in the path leading to the target is noted. From the affected layer on different *path chunks* are built and retrieved. Therefore, the *menudepth* slot holds the assumption that after an error in the path is noted, the erroneous (old) *path chunks* are used only for the layers that have not changed. The *menudepth* chunk furthermore holds knowledge about which layer is the final layer for each target. The *errorpath* slot in the goal buffer holds the knowledge about an occurred change in the path leading to any target; it is not reset between different targets. The *finaltarget* and the *subtarget* slot hold the target and the subtarget as a string. These two slots are used to determine if the target has been found and also required for a superficial visual search utilized on the last page and for researching a subtarget.

Mechanisms en-detail

Initiation In the beginning of each run, the production *start* requests a *meaning chunk*. The building of a *path chunk* is initiated. The production *meaning-in-goal* then copies the retrieved meaning of the target into the slot *target-meaning* of the chunk in the goal buffer and into the *target-im* slot of the chunk in the imaginal buffer. Then, a retrieval request for a *path chunk* leading to the target is initiated with variations[2] of the production *look-for-path.*

Association approach Without prior experience with the specific target, a *path chunk* leading to the target is not retrievable and the production *change-strategy* fires, followed by *find-word* and *reading-word*. Visual attention is directed to the top of the page (to the highest location bellow the current visual attended location) and this location is visually processed. Variations[3] of the production *process-word* then visually encode the current word. For all layers, except the last layer, a request for a *meaning chunk* holding the meaning of the current word is initiated. If such a chunk is found the production *searching association* then initiates the search for an *association chunk* containing the current word and the target. If such an *association chunk* cannot be found, the production *no-association-found* clears the visual buffer and the search continues with the production *find-word*. If an *association chunk* is retrieved, variations[4] of the

[1] The model can be downloaded at https://depositonce.tu-berlin.de/handle/11303/5548.

[2] The variations of *look-for-path* consider two aspects: First, whether or not there was an error in the path and second, the differences in menu depths. This ensures that for a detected change in menu depth the old (misleading) path is not retrieved.

[3] There are variations of *process-word* for the last layer and for the other layers except the last. The variations consider the value of errorpath slot in the goal buffer and the value of the count in the imaginal buffer.

[3] Variations of *association- found* depend on the value of the count slot of the imaginal buffer.

production *association-found* update the *path chunk* in the imaginal buffer. If, for example the first subtarget in path is found, then the value of the *count slot* in the *imaginal buffer* is changed from *one* to *two*. Furthermore, the *first slot* of the *path chunk* is filled with the current *subtarget*, which is also copied into the *subtarget slot* of the goal buffer. A cursor move is initiated and the productions *prepare-click* and *click* initiate the motor movements to press the button. The productions *waiting-click* or *waiting-last-click* (for the final click, in order to initiate the backing procedure) let the model wait until the *manual buffer* is free. After the manual buffer is free a variation of the production *look-for-path* fires again. For the last layer, the elaborate procedure of reading from top to bottom and searching for *association chunks* is replaced by a superficial visual search procedure. If the word in the visual buffer and the word in the slot *finaltarget* of the chunk in the goal buffer are different, a variation of the production *process-word-last-page-wrong* will fire. Via the production *find-word* the next word is searched. If they are the same a variation[5] of *process-word-last-page-correct* will copy the last slot of the path into the *path chunk* in the imaginal buffer, raise the *count slot* and change the value of the *subarget slot* in the chunk in the goal buffer. A cursor move is initiated and the productions *prepare-click* and *click* press the button with the target.

Path Navigation If a *path chunk* leading to the target is retrieved a variation of the production *found-that-path*, depending on the value of the *count slot* in the imaginal buffer, fires. This production copies the value of the relevant slot (e.g. the *subtarget*) from the *path chunk* in the retrieval buffer into the *path chunk* in the imaginal buffer and changes the *count.* The production *find-location* requests for a *meaning chunk* of the relevant subtarget. The production *found-location* indicates that a location was retrieved and the visual attention is moved to the retrieved location. Then the visual buffer and the *subtarget* slot of the chunk in the goal buffer are compared. If they are the same, then the retrieved location is correct and the production *checking-match* fires, followed by *click-location* and *waiting-click*.

Modified Interfaces In the following subsection an overview on mechanism dealing with the modified interfaces is given, for a detailed description see box 1.

The retrieved location is visual inspected and the subtarget is not found at the retrieved location, either because there is a different word, or no word at the retrieved location. This is indicated by the productions *checking-no-match* or *checking-empty*. A visual search for the subtarget is then initiated with the production r*ead-top-to-bottum-again-2*. Visual attention is directed to the top of the page and the production *scan-1* encodes the visual-location. If the word in visual buffer is the subtarget then *scan-correct* fires otherwise *scan-incorrect* moves the visual attention to the next highest word. If the subtarget is found it is selected via *prepare-click* and *click*. If the visual search via *scan-1* and *scan-incorrect* does not lead to the subtarget and the bottom

1. *Depth changes from three to four layers; the path is different from the third layer on.* An update adds a new layer to all targets of the shopping app, e.g. the old path for *alcohol free beer* is *1.stores 2.drinks 3.alcohol free beer* the new path is *1.stores 2.drinks 3.beer 4.alcohol free beer.* In the third run (after an update), *alcohol free beer is searched* on the third layer. But the retrieved location does not contain *alcohol free beer*. The third layer is rescanned from top to bottom in search for *alcohol free beer,* without success. The value of *errorpath* slot in the goal buffer is changed to true. The third layer is then read from top to bottom and for each word an attempt to retrieve *an association chunk* is made. The model visually encodes *beer* and an *association chunk* is retrieved The third slot of the *path chunk* in the imaginal buffer is assigned the value *beer* and beer is selected. On the fourth layer, the attempt to retrieve a *path chunk* that leads to *alcohol free beer* with the slot four having a value is made. Such a chunk cannot be retrieved. The superficial search approach for the last page will be used and directly search for *alcohol free beer is initiated*. For the next product, navigation is realized with the old *path chunk* for layer one and two. For the third and fourth layer the attempt to retrieve a *path chunk* with four layers will fail. The association approach or for the last page the superficial search will be used. In the fourth run the correct *path chunks* for all layers can be retrieved.

2. *Depth changes from three to four layers; the path is different from the second layer on, mixed list*. The path for lawn in the real-estate application changes depending on preselection. If house is preselected, the path is 1.search, 2.garden, 3.lawn and changes to 1.search 2.more 3.garden 4.lawn if apartment is preselected. On the second layer the subtarget garden is not found at the retrieved location. A rescanning of the page is unsuccessful. As in 1) an errorpath in the goal buffer is noted and the approach is changed to reading from top to bottom and searching for association chunks. A new *path chunk* is then built. For the third and fourth layer the attempt will be made to retrieve a *path chunk* with four layers leading to lawn and then utilize the reading and association approach or the scan approach for the last page. For the next target, the value of the menudepth slot in the goal buffer indicates constant depth; therefore it is not influenced by the error in the path. In the fourth run, a *path chunk* for lawn can be found.

3. *Depth changes from four to three layers; the path is different from the third layer on.* An update removes a layer for all targets from the shopping list app. The path for *alcohol free beer* changes from *1.stores, 2 drinks, 3.beer 4.alcohol free beer* to *1.stores, 2.drinks 3.alcohol free beer.* On the third layer the subtarget *beer* is not found at the retrieved location. The third layer is rescanned from top to bottom in search for *beer* and the target *alcohol free beer is found*. The errorpath slot in the goal buffer is changed to true. The third slot of the *path chunk* in the imaginal buffer gets the value *alcohol free beer* assigned and the target is selected,. For the next product navigation is realized with the old *path chunk* for layer one and two. For the third layer the attempt to retrieve a *path chunk* with three layers will fail. The superficial search will be used for the last page and the building of *path chunks will be completed*. In the fourth run the correct path chunks can be retrieved.

4. *Depth changes from four to three layers; the path is different from the second layer on (special case).* First apartment is preselected. Thus, the path for lawn in the real-estate application is *1.search 2.other 3.garden 4.lawn*. Then house is preselected and the path is *1.search, 2.garden, 3.lawn*.On the second layer, the subtarget other is not at the retrieved location. The page is rescanned and *other* is found and selected. On the third page garden is searched for unsuccessfully. The value of the errorpath slot in the goal buffer is then set to true and the model goes back to the second page. A new *path chunk* is built from the second page on via association chunks and reading top to bottom. For the next target, the value of the menudepth slot in the goal buffer indicates constant depth; therefore, it is not influenced by the error in the path. In the fourth run, a *path chunk* for lawn can be found.

Box 2 : A description on how different changes in the interfaces are processed by the model.

of the page is reached, an *errorpath* will be noted in the goal buffer. An *errorpath* in the goal buffer will also be noted, if the target is found while scanning for a subtarget (via *scanning-path-is-wrong*). After the *errorpath* slot in the goal buffer is set to true and the subtarget has not been found, the production *error-in-path-2* resumes the visual attention to the top of the page and via *find-word, reading-word* and *association-found* a *path chunk* is build in the imaginal buffer. The changed value of the *errrorpath* slot will stay in the chunk in the goal buffer for all further model runs. Therefore, if the *errorpath* value in the goal buffer is set to *true*, a wrong *path chunk will not be retrieved*. Either no *path chunk* is retrieved and navigation is implemented via *association chunks*, or *path chunk* are utilized only as long as they are correct. For example, if the *path chunk* is helpful for layer one and two – navigation with this chunk for these layers is implemented, but different *path chunk* are sought for on the third layer.

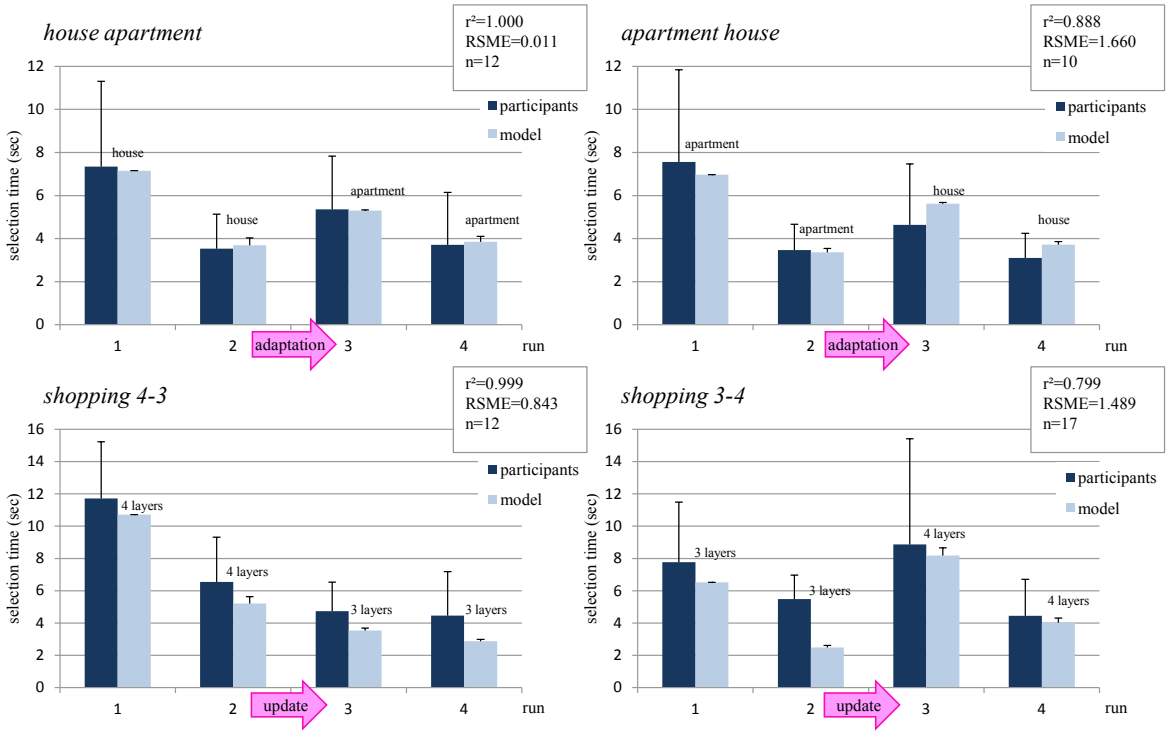


Figure 2: Mean target selection time for the four different studies for the modeled and empirical data.

Results

Data processing

The empirical data comprises of four studies, with a varying number of student participants ($10 < n < 17$). The model was run 10 times per study. Target selection time (for both model and empirical data) is defined as the time between the selection of the target and the selection of the first subtarget. Extreme values were excluded from analysis. Mean target selection time for each target was calculated and averaged over the runs (six to nine targets per run). Mean target selection time for each run was calculated and averaged over the participants. Model and empirical data were compared via goodness of fit indices and qualitative analysis of graphs.

Model parameter

The latency factor (lf) and the retrieval threshold (rt) parameter were fit to match the data of the study house apartment. They are set to the values: lf = 0.1 and rt = -1.5.

Comparison of modeled and empirical data

The model provides a good to very good fit to the data, see figure 2 for the comparison of the mean target selection times of the empirical and modeled data. The main trends of the four studies are mapped in the modeled data, as are most of the absolute values.

1. The *house apartment* study reveals a decrease of mean target selection time followed by an increase and again a decrease. This is exactly represented in the modeled data; $r^2 = 1.0$. The absolute values of the modeled and empirical data are also very close; RSME = 0.011.

2. The *apartment house* study reveals the same pattern (decrease, increase and decrease); $r^2 = 0.888$. The mean target selection time predicted by the model is slightly lower in the first two runs and a bit higher in the last two runs; RSME = 1.660.

3. The empirical and modeled mean target selection time of the *shopping 4-3* study indicate a decrease from run 1 to run 4, with the decrease leveling out towards the last run. The magnitude of the decrease is very similar for both

datasets; r^2= 0.999. In all runs the mean target selection times predicted by the model are lower than those of the participants; RSME = 0.843.

4. For the *shopping 3-4* study, both datasets show a pattern of a decrease, followed by an increase and a final decrease. The modeled increase in mean target selection time between run 2 and 3 is greater than the increase found in the empirical data. The magnitude of the other variations are similar for the empirical and modeled data; r^2 = 0.799. The mean target selection time predicted by the model is lower than that of the participants, especially in the third run; RSME = 1.489.

Summary

In summary, all trends found in the empirical data are predicted by the model, with a high r^2 = [0.799; 1.0] for all four studies. Decreases and increases in target search time are predicted by the model. Therefore, the model provides information on learning and relearning effects for different applications. Moreover, the absolute values are met by the model for the majority of data points, with RSME = [0.011; 1.660], values which are lower than the average STD of the empirical data. Hence, the model can appropriate depict mean user behavior at different time points during a repeated target selection task.

Discussion

The modeling approach provides accurate predictions of the data from four studies with two different apps. Efficiency, learnabilty and the impact of updates or of adaptivty are predicted by the model.

Only a few steps are necessary to alter the model for a different app; world knowledge needs to be provided in form of *association chunks* and reading ability as *meaning chunks*. Furthermore, a compilation of the *targetlist*, containing the targets and of the *menudepthlist*, containing the menu depth of the targets before and after an update, is required.

The model can easily be extended to other list-style hierarchical apps with different menu depths than 3 and 4 layers and to apps with different switches in the number of layers. To do so, variations of the productions *process-word*, *process-word-last-page* and *look-for-path* are required.

Plausibility of Modeling Decision

A general modeling approach facilitates the adaptability of the model. To achieve this, partially simplified assumptions have been made. This applies especially to the *menudepth* slot, which offers a technical solution for the handling of varying menudepth. This slot contains meta-knowledge of the model, in other words knowledge about what kind of update occurred. The *menudepth* slot holds information if the update relates to the entire menu structure or if it relates to individual paths. Furthermore, the *menudepth* slot is useful to identify the last page. It is plausible to assume that users know if the current page is the last page. However, users are likely to obtain this knowledge with the help of visual features. Since these in turn significantly differ between apps, it is useful for a general approach to detect the last page in a simplified manner.

The reading direction of the model is always directed from top to bottom and each item is processed, this is a further simplification. It is possible to model visual processing of menus more precisely (Bailly, Oulasvirta, Brumby and Howes, 2014). Since, the goal of our work was to predict average mean search time for items, our chosen simplification of visual processing was sufficient.

Another simplified assumption is, that the imaginal holds the count of the current page. This is done only for practical reasons to make model steering easy and adjustable to different updates. Furthermore, it does not affect overall item search time.

Limitations and Further Steps

To use the model approach for usability testing, a remaining obstacle is the need of prototyping the interface. In order for the ACT-R model to interact with the application, a copy of the app in Lisp is required. To avoid this effortful step, we are working on a tool called ACT-Droid (Doerr, Russwinkel, & Prezenski, 2016).With ACT-Droid android apps can be connected directly with ACT-R models. This tool is will reduce the practical hurdle of testing usability of apps with cognitive models further. Further steps to improve the model are to replace the mouse movements in the model with touch movements, as provided by ACT-Touch (Greene & Tamborello, 2013). Besides, an in-depth validation of the model with eye tracking data is planned. A new study should also investigate if the model can predict average click times for each menu layer as well. The empirical data of the current studies, do not allow such predictions, due to the study design. Nevertheless, on an individual level the model and empirical data show, that after an update (shopping app) and after an unexpected adaption (real-estate app) search time increases.

We are also intending to look into how far mechanisms of the current model can be used to develop a model for hierarchical apps with icons instead of text.

Moreover, our general modeling approach for hierarchical list-style apps is not only useful for such apps. It is well suited to predict the average user search time for all kinds of list-based interfaces that spread semantically related subtargets across multiple layers.

References

Bailly, G., Oulasvirta, A., Brumby, D. P., & Howes, A. (2014). Model of visual search and selection time in linear menus. *Proc. of the 32nd Annual ACM Conference on Human Factors in Computing Systems - CHI '14, (pp.* 3865–3874)**.** Toronto, Canada: ACM.

Byrne, M. D. (2001). ACT-R/PM and menu selection: applying a cognitive architecture to HCI. *Int. J. Human-Computer Studies*, 55, 41-84.

Das, A., & Stuerzlinger, W. (2007). A cognitive simulation model for novice text entry on cell phone keypads. In *Proc. of the 14th European Conference on Cognitive Ergonomics*, (pp.141-147). London, UK: ACM.

Doerr, L.-M., Russwinkel, N., & Prezenski, S. (in submission). ACT-Droid: ACT-R Interacting with Android Applications. In D. Reitter & F. Ritter (Eds.), In *Proc. of the 14th Int. Conference on Cognitive Modeling.* Pennsylvania, USA.

Greene, K. K. & Tamborello, F. P. (2013). Initial ACT-R extensions for user modeling in themobile touchscreen domain In *Proc. of the 12th Int. Conference on Cognitive Modeling.*(pp.348-353)Ottawa, Canada.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *Proc. of CHI´04*, (pp. 455–462). New York, USA: ACM.

Prezenski, S., Lindner, S., Moegele, H., & Russwinkel, N. (in preparation). Archetyping the User.

Prezenski, S., & Russwinkel, N. (2014). Combining Cognitive ACT-R Models with Usability Testing Reveals Users Mental Model while Shopping with a Smartphone Application. *Int. J. on Advances in Intelligent Systems*, 7(3), 700–715.

St.Amant, R., Horton, T.E., & Ritter, F.E. (2007). Model-based evaluation of expert cell phone menu interaction. *Transactions on Computer-Human Interaction*, 14(1), 1-14.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Modeling phonological similarity effects on the self-organization of vocabularies

Javier Vera (jxvera@gmail.com)
Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez
Santiago, Chile

Abstract

This work describes the formation of a language under phonological similarity constraints. Classical studies on human subjects show that in recalling experiments short-term memory performance was impaired for phonologically similar words versus dissimilar ones. Here, artificial individuals confound phonologically similar words according to a predefined parameter: the more the parameter, the more the confusion. Theoretical results present proofs of convergence for a particular case of the model within a worst-case complexity framework. Computer simulations describe the evolution of an energy function that measures the amount of local agreement between individuals. The main finding is the appearance of sudden consensus changes at critical parameters.

Keywords: Phonological similarity; Short-term memory; Automata networks; Vocabulary.

Introduction

In natural language, a typical linguistic interaction is not a simple sequence of individual actions, but also a form of *joint activity* that involves cooperation and coordination between participants (Tomasello, 2008). What is more, inside the dialogue language users tend to converge in their choice of constructions. This mutual convergence process, or *alignment*, has been extensively studied within computational studies of the formation of language through the Naming Game (Steels, 1995, 2011; Baronchelli, Felici, Caglioti, Loreto, & Steels, 2006; Loreto, Baronchelli, Mukherjee, Puglisi, & Tria, 2011). It attempts to ask how on a population of agents only from local interactions it arises a shared word-meaning association (the simplest version of a *vocabulary*). This model considers a finite population of agents, where each one is endowed with a memory in which it stores an in principle unlimited number of words. At each discrete time step, a pair of agents is selected: one plays the role of *speaker*, one plays the role of *hearer*. First, the speaker refers to an object by using a word. Next, the hearer tries to identify the referent. For this purpose, the hearer inspects its own memory: (1) if the word belongs to the memory of the hearer, both speaker and hearer cancel all the words in their memories, except such word; or (2) if the word does no belong to its memory, the hearer adds the word to its memory. The Naming Game only is possible if pairwise interactions are a kind of joint activity. In a more realistic scenario, both agents interact in a context where the object is located and share focus on the object by means of pointing or eye-gazing. The interaction continues until both agents reach a common word associated to the object, a word-meaning association.

Here, it is assumed that the development of word-meaning associations is founded on self-organization mechanisms arising only from *local* interactions between agents (Steels, 1995, 1996; Baronchelli et al., 2006). Given this self-organized nature, Automata Networks (AN) (Neumann, 1966; Wolfram, 2002) provide the adequate framework to explore alignment from a computational (and mathematical) point of view. AN are extremely simple models where each vertex of a network evolves following a local rule based on the states of "nearby" vertices. Despite of the simplicity of the defining rules, AN exhibit astonishing rich patterns of behavior. Thus, a word-meaning association can be considered as a complex pattern.

What is the relationship between cognitive mechanisms of the individuals and the emergence of language on artificial populations? The question entails the study of computational machines which exhibit limitations and constraints of *human* language users. This work stresses a novel question related to models of the self-organization of language: To what extent do working memory constraints influence the alignment of shared conventions on artificial populations of agents? Language users (and therefore agents playing computational games of the formation of language) suffer limited short-term memory constraints (A. Baddeley & Hitch, 1974; A. Baddeley, 2007). Particularly, *phonological similarity* effects suppose that individuals confound word items sharing large portions of phonological content. A classical work (A. D. Baddeley, 1966) reports experiments where subjects heard sequences of unrelated words and tried to recall in the correct order. The results suggest that memory performance was impaired for phonologically similar words (man, cad, mat, cap, can) versus dissimilar ones (pen, sup, cow, day, hot). This effect is a strong evidence for the existence of the *phonological loop*, understood as part of the short-term (working) memory system (A. Baddeley & Hitch, 1974; A. Baddeley, 2007).

This paper does not attempt to develop a "realistic" model, but rather an abstract symbolic approach that extracts the essential elements of the problem. The simulations described here are based on a parameter that measures the amount of phonological confusion between words (considered as a way to describe the influence of working memory limits): the more the parameter, the more the confusion. The work explores the hypothesis of there being a critical range of the parameter that implies drastic changes in the shared conventions of the entire population. Therefore, the features of language (in particular, the consensus on word-meaning associations) emerge abruptly at some critical range of phonological confusion (Hauser, 1996). This hypothesis is strongly related to previous work on on the absence of stages in the formation and evolution of human languages (Bickerton, 1998; Calvin & Bickerton, 2001; Ferrer-i-Cancho & Solé, 2003).

The work is organized as follows. Section "Description

of the model" explains basic notions on AN, the instrumentalization of phonological similarity and the rules of interaction. Section "Theoretical issues" reports simple mathematical results related to the convergence of particular cases of the model. The next section ("Simulations") describes simulation tasks based on an energy operator, over a parameter that measures the amount of similarity confusion between words. A brief discussion of the results is presented in the final section.

Description of the model

Elements of the AN model

Roughly speaking, the AN model involves the following elements:

- A regular grid graph: vertices represent individuals; edges represent possible communicative interactions. More generally, as in the section "Theoretical results", the graph can be a connected, undirected and simple network.
- Each individual is associated to a *state* that eventually changes along time. This *state* is a way to represent the language of the individual at some time frame.
- A set of *local rules* that define how the system changes. The local rule associated to one individual considers as inputs the states of the nearby individuals (the *neighbors*).
- A function, the *updating scheme*, that indicates the order in which the individuals are updated. Two updating schemes are considered in this work: (1) the *fully-asynchronous* scheme, where at each time step one individual is choosen uniformly at random; and (2) the *sequential* scheme, defined as a permutation of the set of vertices.

In what follows, some of the previous elements will be treated in greater depth.

Basic notions

The set $P = \{1, ..., n\}$ represents the population of individuals, located on the vertices of the regular grid graph $G = (P, E)$, where E is the set of edges. The vertex $u \in P$ only interacts (or "talks") with the set of adjacent vertices $V_u = \{v \in P : (u, v) \in E\}$ (the *neighborhood*). Each individual considers four neighbors: up, down, left and right ones (*Von Neumann* neighborhood). The individual $u \in P$ is endowed with a *memory* set M_u in which it stores words belonging to a finite set W, with p elements.

Let $\Sigma = \{a_1, ..., a_s\}$ be a set of sounds. Each word of W is constructed by a combination of sounds taken from Σ. For instance, $a_3a_1a_8 \in W$. The *length* of the word x is its number of sounds. For the sake of simplicity, all the words have the same length L. $x(k)$, with $k \leqslant L$, denotes the $k-$th sound (or position) of the word x. To explicitly measure the amount of phonological similarity between words, the *Hamming distance* $H(x, y)$ between the words x and y is defined as the number of positions in which they differ. Consider two words $a_4a_6a_5$ and $a_7a_6a_3$, then $H(a_4a_6a_5, a_7a_6a_3) = 2$.

Confusion parameter

To explicitly measure the ability to distinguish between words, the *confusion parameter* $\varepsilon \in [0, 1]$ is defined. Suppose that the vertex u faces two words x, y. Then,

if $H(x, y) > \varepsilon L$, u **distinguishes** the words x and y

else u **confounds** the words x and y (or simply "$x = y$")

For instance, the individual $u \in P$ is confronted with the words $x = a_1a_8a_6a_4$ and $y = a_1a_8a_6a_3$. Two values of ε are considered, 0 and 0.5:

- ($\varepsilon = 0$) $H(x, y) = 1 > \varepsilon L = 0 \times 4 = 0$, then u distinguishes the words x and y
- ($\varepsilon = 0.5$) $H(x, y) = 1 < \varepsilon L = 0.5 \times 4 = 2$, then u confounds the words x and y

Local rules

Inspired in the Naming Game (Baronchelli et al., 2006), the *local rule* associated to the individual $u \in P$ is based on two possible actions on the memory M_u:

- u updates its memory M_u by the **addition** of words; or
- u **collapses** its memory if M_u is updated by cancelling all its words, except one of them.

Both actions attempt to take into account *lateral inhibition* strategies (Steels, 1995, 2011) in the alignment process: the individuals add words in order to increase the chance of future agreements (local consensus), and defect the words that do not cooperate with mutual understanding. In a *collapse*, the individuals prefer the *minimal* word, according to the *lexicographic order* over the set of words. The *lexicographic order*, denoted $\prec$, is a generalization of the typical alphabetical order of words on the alphabetical order of their component letters (or sounds). For example, in the dictionary the word "Me" appears before "My" because the letter e comes before the letter y in the alphabet. In some sense, the word "Me" is *lower* than the word "My". Formally, the order $<$ is defined on the set Σ. Two words x and y of length L are considered. Then, $x \prec y$ (x is *lexicographically* lower than y) if the first position in which they differ, say $k \leqslant L$, satisfies $x(k) < y(k)$. For instance, given the set of sounds $\Sigma = \{a, b, c, d\}$, with $a < b < c < d$, the words *abc*, *bcd* and *cda* satisfy $abc < bcd < cda$. Therefore, *abc* is the *minimal* word or, in other terms, $\min(\{abc, bcd, cda\}) = abc$. Associated to the previous words "Me" and "My", it is possible to write $\min(\{\text{"Me"}, \text{"My"}\}) = \text{"Me"}$.

The preference for the minimal words can be viewed in accordance with the following hypothetical scenario (Nowak & Krakauer, 1999). It is possible to think in a population of early hominids for which leopards represents a higher risk than cows. So, the word "leopard" may be more valuable than "cow". In the terms of this paper, "leopard" can be the *minimal* word.

At time step t, the vertex $u \in P$ is selected according to the updating scheme (*fully asynchronous* or *sequential*). Consider a simple population $P = \{1,2,3,4,5\}$ (each number represents one individual). For a *fully asynchronous* scheme, at each time step any individual of P can be selected (for instance, the individual 3). In the next step, the individual 3 can be selected again. For a *sequential* scheme, a permutation of the set P is defined, for instance, the order 5-4-3-2-1. The individual 5 updates first, then the individual 4 updates, taking into account the effects of the changes in the first individual, and so on. At time step 6, the dynamics starts in the same previous way. In other terms, a *sequential* scheme supposes that each individual is updated after that all the other individuals of the population have been updated.

The individual $u \in P$ is completely characterized by its *state* (M_u, x_u), where M_u is the memory to stores words (M_u is a subset of W) and $x_u \in M_u$ is a word that u conveys to the vertices of V_u. The model induces specific communication roles: the vertex u plays the role of "hearer" (it receives the words conveyed by its neighbors); the neighbors of u play the role of "speaker" (they convey words to the vertex u). The set of all words conveyed by the speakers can be re-written as two subsets: N_u and B_u. Roughly speaking, N_u includes the unknown words, and B_u includes the known ones.

The state pair (M_u, x_u) changes according to the following steps, which define the *local rule* of the automata (see Fig. 1):

step 1 the vertex u defines two sets:

$$N_u = \{x_v : (v \in V_u) \wedge (\forall y \in M_u, H(x_v, y) > \varepsilon L)\}$$
$$B_u = \{x_v : (v \in V_u) \wedge (\forall y \in M_u, H(x_v, y) \leqslant \varepsilon L)\}$$

step 2

if $N_u \neq \emptyset$, M_u **adds** the words of N_u
else M_u **collapses** in the word $\bar{x}$, selected at random from the set $\{x \in B_u : H(x, \min(B_u)) \leqslant \varepsilon L\}$ (that is, the new state is $(\{\bar{x}\}, \bar{x})$)

Step 1 comprises (1) the speaker's behavior (the neighbors convey words to the vertex u); and (2) the definition by the hearer of the sets N_u and B_u. Given a conveyed word x_v, $v \in V_u$, the hearer u decides between: either x_v is added to N_u if for all $y \in M_u, H(x_v, y) > \varepsilon L$; or x_v is added to B_u, otherwise.

Step 2 summarizes the behavior of the hearer in order to *align* itself with the speakers. In the case that $N_u \neq \emptyset$, the hearer simply adds to its memory the words of N_u. Otherwise ($N_u = \emptyset$), the hearer collapses its memory in the word $\bar{x} \in B_u$. The word $\bar{x}$ is selected uniformly at random from $\{x \in B_u : H(x, \min(B_u)) \leqslant \varepsilon L\}$. Thus, the preferred word $\min(B_u)$ can be confused by "similar" ones (with $H(\cdot, \min(B_u)) \leqslant \varepsilon L$).

Dynamics of the automata

As initial configuration each individual receives a word constructed by a random combination of L sounds from the set of symbols Σ. Thus, at $t = 0$ each individual is associated to a state of the form $(\{x\}, x)$, with $x \in W$. Each discrete time step $t \geqslant 0$ supposes that one individual, say u, is selected uniformly at random (the *fully-asynchronous* scheme) or according to a permutation of the set of vertices (the *sequential* scheme). The individual receives the words conveyed by its neighbors: it plays the role of hearer, the neighbors play the role of speaker. Regarding to the conveyed words the individual follows the two defined steps in order to decide possible changes in its own language state:

step 1 the individual defines the sets N_u (unknown words) and B_u (known words).

step 2 the individual either **adds** words if N_u is non empty or **collapses** its memory in the minimal word of B_u, if N_u is empty.

Both steps involve the possibility of confusion between *similar* words. In the next time step $t+1$ another individual (possibly the same one) is selected, and so on.

A *fixed point* is a configuration invariant under the application of local rules, which can be interpreted as a final consensus configuration, where all individuals agree about some linguistic convention.

Theoretical issues

Convergence under $\varepsilon = 0$

An interesting problem related to the formation of consensus on linguistic conventions is to propose a proof of convergence. Given the mathematical framework of this paper, the problem becomes to count (in the *worst* case) the number of simulation steps whom the dynamics needs to stop, that is, to prove that after a finite number of time steps the population reaches a shared word-meaning association. Despite that other works have solved related tasks (DeVylder & Tuyls, 2006), the novelty of the rest of this section is to develop a convergence proof based on the worst-case complexity, which measures the amount of resources (running time) needed by the system if it is considered as an *algorithm*. Running time, defined as the number of steps until the entire population reaches a global consensus language, indicates the largest dynamics performed by the automata given the size n of the population (denoted $O(f(n))$, where $f(n)$ is a function of n). For instance, $O(n^2)$ means that in the worst-case the running time has a growth rate scaling as n^2.

In this section, individuals are located on a general undirected and connected network (not necessarily a regular *grid*), and they do not confound words, that is, $\varepsilon = 0$.

It is straightforward to notice that at $\varepsilon = 0$ $(\forall y \in M_u, H(x_v, y) > \varepsilon L)$ is equivalent to $(x_v \notin M_u)$. Roughly speaking, the expression $(\forall y \in M_u, H(x_v, y) > \varepsilon L)$ means that x_v is "different" to every word in M_u and, therefore, $x_v \notin M_u$. As a consequence, the two steps of the rule take a simpler form:

step 1 the vertex u defines two sets:

$$N_u = \{x_v : (v \in V_u) \wedge (x_v \notin M_u)\}$$

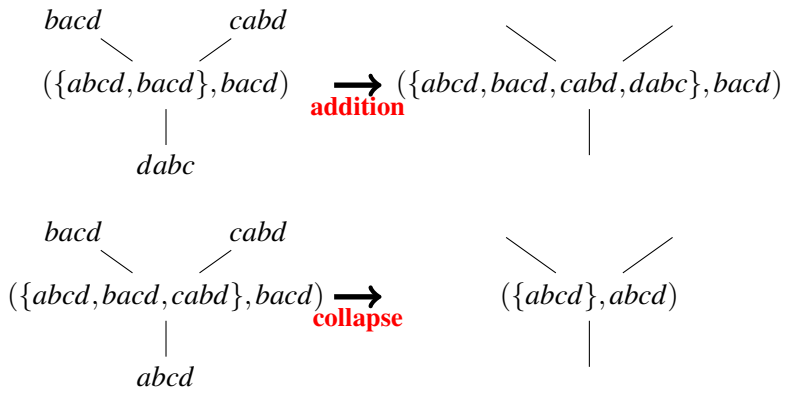


Figure 1: **Example of the two actions at $\varepsilon = 0$.** The order $a < b < c < d$ is defined on the set $\Sigma = \{a,b,c,d\}$. $W = \{abcd, bacd, cabd, dabc\}$. The *lexicographic* order establishes $abcd \prec bacd \prec cabd \prec dabc$. The confusion parameter is set to $\varepsilon = 0$ (that is, the individuals do not confound words). The central vertex ($u \in P$) has been choosen. Four individuals participate in the interaction: the central vertex u and its three neighbors of V_u. Two different configurations are showed. First row **(addition)**: $B_u = \{bacd\}$ and $N_u = \{cabd, dabc\}$. Second row **(collapse)**: $B_u = \{abcd, bacd, cabd\}$ and $N_u = \emptyset$.

$$B_u = \{x_v : (v \in V_u) \wedge (x_v \in M_u)\}$$

step 2

if $N_u \neq \emptyset$, M_u **adds** the words of N_u
else M_u **collapses** in the word $\min(B_u)$ (the new state is $(\{\min(B_u)\}, \min(B_u))$)

First of all, a sequential updating scheme is considered. This means that a *permutation* of the vertices is defined. As previously noted, each individual is updated after that all the other individuals of the population have been updated. In the worst case, each time step supposes that one individual adds one word. This process ends when some individual has all possible words, that is, after $p-1$ steps (there are p words). Since the population has size n, after $n(p-1)$ steps the individuals must collapse their memories. Then, at step $t^* = n(p-1)$ all individuals have been collapsed at least once. As the next theorem shows in detail, the minimum conveyed word at t^* propagates in the system until it reaches a fixed point where all individuals convey this minimum word.

Theorem 1 *(Goles, Montealegre, & Vera, 2016)*
Consider a population of n individuals playing the automata model with the set of p words W and confusion parameter $\varepsilon = 0$. Then,

(I) for the sequential scheme, the system converges to fixed points in at most $O(n^2 p)$ steps;

(II) for the fully-asynchronous scheme, the system converges to a fixed point in expected time $O(n^2 p \log(n))$.

Proof 1 (I) Initially there are p words. Then, in at most $p-1$ updates a vertex has collapsed for the first time (in the worst case the vertex must add every possible word, one at a time). This implies that in $n(p-1)$ steps ($p-1$ updates of each vertex) all vertices have collapsed at least one time. Let m be the minimum conveyed word at step $t^* = n(p-1)$, and let u be a vertex such that $x_u = m$ (in more precise terms, $m = \min(\{x_u\}_{u \in P})$).

Since u has collapsed at least one time, the updating scheme is sequential, and m is the minimum word, then u must have another neighbor $v \in V_u$ conveying m. In consequence, after t^* both vertices u and v will remain conveying m, and at each time a neighbor of any of these two vertices will necessarily collapse in the word m. The graph has a diameter of $O(n)$ and each np steps it occurs a collapse. Therefore, in at most $O(n^2 p)$ steps the system converges to a fixed point where all vertices convey the same word m.

(II) In the fully-asynchronous scheme, at each time step a single vertex is picked independently and uniformly at random. The expected convergence time grows by a factor of $O(\log(n))$ with respect to the sequential scheme. Notice that in the proof of the part (I) it is shown that after updating [$O(np)$ times] [each vertex], a fixed point is reached. From the well known coupon collector's problem (see, for instance, (Grimmett & Stirzaker, 2001)), the expected number of steps required to update [at least once][every vertex in the graph (or pick every coupon)] is $O(n \log(n))$. Since in a fully asynchronous updating scheme each step is independent to the others, the result follows.

A brief note on dynamics under $\varepsilon = 1$

At $\varepsilon = 1$, individuals confound any pair of words $x, y \in W$. Indeed, for all $x, y \in W$ $H(x,y) \leqslant \varepsilon L \leqslant L$. A simple result says:

Proposition 1 *Consider the automata model with confusion parameter $\varepsilon = 1$. Suppose that the individual $u \in P$ has been selected at some time step. Then,*

1. $N_u = \emptyset$; *and*
2. $B_u \neq \emptyset$

A relevant consequence of Proposition 1 is that **additions** are not allowed. This fact requires further work.

Simulations

Protocol

To describe the amount of agreement between individuals, an "energy" operator is defined. This energy-based approach arises from a "physicist" interpretation, related to (Regnault, Schabanel, & Thierry, 2009): the energy measures the amount of local unstability of the system. At each neighborhood V_u, the function $\sum_{v\in V_u} H(x_u,x_v)$ is defined, which measures the *Hamming* distance between the word x_u and the words conveyed by the neighbors of the individual u. This function is bounded by 0 (in case of *agreement*, that is, the individual u and its neighbors convey the same word) and $4L$ (*disagreement*, which means that the individual u and its neighbors convey radically different words). Summing over all individuals defines the total energy function at some time step t:

$$E(t) = \frac{1}{4Ln} \sum_{u\in P} \sum_{v\in V_u} H(x_u,x_v) \qquad (1)$$

The function $E(t)$ is bounded: $0 \leqslant E(t) \leqslant 1$. Thus, the dynamics of the AN model can be understood as the trajectory between initial configurations associated to large amounts of unstability (with $E(0)\sim 1$) and final consensus configurations where $E(t)\sim 0$, or equivalently, all individuals convey similar -in the sense defined by the *Hamming* distance- words.

The analysis focuses on two-dimensional periodic lattices of size $n = 128^2$ with Von Neumann neighborhood. The simulations describe $500n$ steps of the energy function $E(t)$. At each time step, one vertex (playing the role of hearer) is selected uniformly at random (*fully-asynchronous* scheme). The plots show average values over 20 initial conditions. An initial condition is defined as follows: each individual receives uniformly at random a word constructed by a random combination of L sounds from the set of symbols Σ, $|\Sigma| = 10$. Word-length L varies from: $\{2,4,8,16,32,64\}$. ε is varied from 0 to 1 with an increment of 10%.

Given the worst-case complexity approach to theoretical aspects of convergence, it is important to notice that on low-dimensional lattices it seems hard that one individual adds $O(np)$ words after it collapses, because lattices are low-connected. This intuition suggests that running times on lattices will be lower than theoretical bounds ($O(n^2 p\log(n))$ for the *fully-asynchronous* scheme).

Results

There are several remarkable aspects, as shown in Fig. 2 and Fig. 3. First of all, $E(t=500n)$ versus ε exhibits at $\varepsilon = 1$ a maximum which is close to 0.5 for all values of L (Fig. 2). Secondly, to the extent L grows, $E(t=500n)$ versus ε evolves more "smoothly": $L \leqslant 8$ supposes "ladder" steps which mean that different values of the confusion parameter ε lead to the same energy. Finally, focusing the description on $L = 32$ (Fig. 3), it is interesting to notice that at $\varepsilon = 0.7$ the average value of $E(t)$ stops approximately after $t = 200n$ steps. This fact exhibits that only a few set of runs does not converge to the global minimum of the $E(t) = 0$. This strongly suggests the appearance of a critical parameter $\varepsilon^* \sim 0.7$ which clearly defines two phases in the evolution of $E(t = 500n)$ versus ε: (1) $\varepsilon < \varepsilon^*$ implies the convergence to the global minimum $E(t) = 0$, where all individuals convey the same word; and (2) for $\varepsilon \geqslant \varepsilon^*$ the dynamics changes drastically until it reaches local minima of the energy function ($E(t) \to 0.5$).

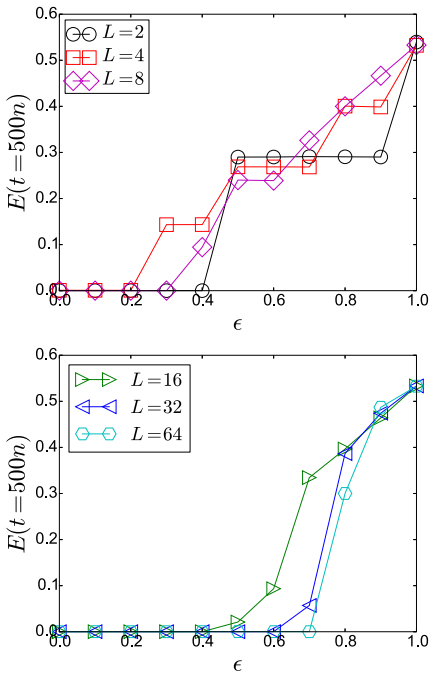


Figure 2: $E(t = 500n)$ **versus** ε. On a two dimensional grid of size $n = 128^2$, the figure shows the final value of the energy function versus the parameter ε, after $500n$ steps or until they reach the global minimum $E(t) = 0$. The plots show averages over 20 initial conditions: for an initial condition, each vertex receives a word constructed by a random combination of L sounds from the set of symbols Σ, $|\Sigma| = 10$. Word-length varies from: $\{2,4,8\}$ (top); and $\{16,32,64\}$ (bottom). ε is varied from 0 to 1 with an increment of 10%.

Discussion

This work summarizes an AN approach to the formation of linguistic conventions under phonological similarity (and, in general, short-term memory) mechanisms. The paper presents the evolution of an energy functional, defined as a word "confusion" average, during the alignment game. Two aspects are remarkable. On the one hand, the appearance of drastic transitions can be related to previous works that focus on the absence of stages in the formation and evolution of human languages (see, for instance, (Bickerton, 1998; Calvin & Bickerton, 2001; Ferrer-i-Cancho & Solé, 2003)).

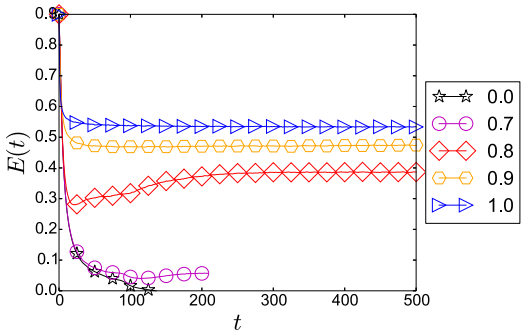


Figure 3: $E(t)$ **versus** t**,** $L = 32$**, for different values of** ε. On a two dimensional grid of size $n = 128^2$, the figure shows evolution of the energy function $E(t)$ versus t, for different values of the parameter ε. Simulations run $500n$ steps or until it reaches the global minimum $E(t) = 0$. The plot shows averages over 20 initial conditions: for an initial condition, each vertex receives a word constructed by a random combination of L sounds from the set of symbols Σ, $|\Sigma| = 10$. $L = 32$ and ε is varied from $\{0, 0.7, 0.8, 0.9, 1\}$.

On the other hand, the proposed model becomes an alternative (mathematical) framework for agent-based studies on language formation.

As a first approach to the convergence of the formation of linguistic conventions, this paper presents within an AN account simple results of the number of steps needed to reach fixed points. As the main tool, the proofs are based on the *worst* case of convergence.

Many extensions of the proposed model should be studied with the purpose to describe the role of cognitive constraints on the formation of word-meaning associations. First, within a mathematical point of view it seems interesting to explore convergence bounds on regular lattices. Also, a comparison between theoretical and numerical convergence times should be carried out. Second, AN allow to study new aspects of the formation of linguistic conventions. Indeed, the model can be extended to other *updating schemes*, for example, the *synchronous* one, where at each time step all individuals are updated. Third, the results should be compared with larger word lengths (L) and several number of symbols (Σ). Finally, future work should involve more realistic ways to measure the amount of phonological confusion and its effects on the formation of conventions.

Acknowledgments

The author likes to thank CONICYT-Chile under the Doctoral scholarship 21140288.

References

Baddeley, A. (2007). *Working memory, thought, and action*. Oxford University Press.

Baddeley, A., & Hitch, G. (1974). Working memory. In G. Bower (Ed.), *Recent advances in learning and motivation* (Vol. 8, p. 47-90). Academic Press.

Baddeley, A. D. (1966). Short-term memory for word sequences as a function of acoustic, semantic and formal similarity. *Quarterly Journal of Experimental Psychology*, *18*(4), 362-365. (PMID: 5956080)

Baronchelli, A., Felici, M., Caglioti, E., Loreto, V., & Steels, L. (2006). Sharp transition towards shared vocabularies in multi-agent systems. *J. Stat. Mech.*(P06014).

Bickerton, D. (1998). Catastrophic evolution: The case for a single step from protolanguage to full human language. In J. R. Hurford, S. M. Kennedy, & C. Knight (Eds.), *Approaches to the evolution of language: Social and cognitive bases* (pp. 341–358). Cambridge: Cambridge University Press.

Calvin, W., & Bickerton, D. (2001). *Lingua ex machina: Reconciling darwin and chomsky with the human brain*. MIT Press.

DeVylder, B., & Tuyls, K. (2006). How to reach linguistic consensus: A proof of convergence for the naming game. *The Journal of Theoretical Biology*, *242(4)*, 818–831.

Ferrer-i-Cancho, R., & Solé, R. V. (2003). Least Effort and the Origins of Scaling in the Human Language. *Proceedings of the National Academy of Science (USA)*, *100*, 788–791.

Goles, E., Montealegre, P., & Vera, J. (2016). Naming game automata networks. *Journal of Cellular Automata*. (accepted for publication)

Grimmett, G., & Stirzaker, D. (2001). *Probability and random processes*. OUP Oxford.

Hauser, M. (1996). *The evolution of communication*. MIT Press.

Loreto, V., Baronchelli, A., Mukherjee, A., Puglisi, A., & Tria, F. (2011). Statistical physics of language dynamics. *Journal of Statistical Mechanics: Theory and Experiment*, *2011*(04), P04006.

Neumann, J. von. (1966). *Theory of self-reproducing automata*. Champain, IL: University of Illinois Press.

Nowak, M. A., & Krakauer, D. C. (1999). The evolution of language. *Proceedings of the National Academy of Sciences*, *96*(14), 8028-8033.

Regnault, D., Schabanel, N., & Thierry, E. (2009). Progresses in the analysis of stochastic 2d cellular automata: A study of asynchronous 2d minority. *Theoretical Computer Science*, *410*(4749), 4844 - 4855.

Steels, L. (1995). A self-organizing spatial vocabulary. *Artificial Life*, *2*(3), 319-332.

Steels, L. (1996). Self-organizing vocabularies. In *Proceedings of artificial life v, nara, japan* (pp. 179–184). Nara, Japan.

Steels, L. (2011, December). Modeling the cultural evolution of language. *Physics of Life Reviews*, *8*(4), 339-356.

Tomasello, M. (2008). *The origins of human communication*. MIT Press.

Wolfram, S. (2002). *A new kind of science*. Wolfram Media.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Spiking Neural Model of Supervised Learning in the Auditory Localization Pathway of Barn Owls

Michael O. Vertolli (michaelvertolli@gmail.com)[1]

Terrence C. Stewart (tcstewar@uwaterloo.ca)[2]

[1]Institute of Cognitive Science, Carleton University, 1125 Colonel By Drive, Ottawa, ON K1S 5B6, Canada
[2]Center for Theoretical Neuroscience, University of Waterloo Waterloo, ON N2J 3G1, Canada

Abstract

We propose a large-scale system, with minimal global topological structure, no local internal structure, and a simple online biologically plausible local learning rule that captures supervised learning in the barn owl. We outline how our computational model corresponds to both the underlying neuroscience and the experimental paradigm used in the relevant prism studies of the barn owl. We show that our model is able to capture the basic outcomes of this experimental research despite learning the initial tuning curves, which is not done in other computational models, and a much more restricted time frame relative to the original experimental condition. We outline some variations between our model and the neuroscience outcomes and suggest future extensions in terms of larger models and time frames, more detailed analyses of the learning parameters, and richer model designs.

Keywords: supervised learning, spiking neural network, auditory localization, barn owl

Introduction

Supervised learning in the brain is generally viewed as a learning episode where one neuron ensemble acts as an instructive signal that modulates connectivity and activity in another neuron ensemble (Knudsen, 1994). The auditory localization pathway in the brain has been used as a model system for studying how the brain performs supervised learning both because experience can alter auditory localization and because the pathways are relatively well studied (Knudsen, 2002). The species that has been studied most extensively in this context is the barn owl (*Tyto alba*; Knudsen, Blasdel, & Konishi, 1979).

There are a number of computational models that have been used to compliment the neuroscience research on supervised learning in the barn owl auditory localization pathway (D'Souza, Liu, & Hahnloser, 2010; Fischer, Anderson, & Peña, 2009; Huo & Murray, 2009; Huo, Murray, & Wei, 2012; Witten, Knudsen, & Sompolinsky, 2008). All of these models are small in scale (under 100 neurons), use highly structured systems that are determined by the modelers, and a rather complex learning rule (usually spiking-time dependent plasticity). We propose a larger-scale (and scalable) system, with minimal global topological structure, no local internal structure, and a simple online biologically plausible local learning rule that captures supervised learning in the barn owl.

Neuroscience Background

The optic tectum of the barn owl (analogous to the superior colliculus in humans) hosts the neurophysiological associations between the auditory localization cues and locations in visual space (Knudsen, 2002). Locations in the optic tectum respond maximally to auditory or visual stimuli located at a specific region of space or the receptive field. The associated neurons are tuned to auditory localization cues that correspond to visual field locations (Brainard, Knudsen, & Esterly, 1992).

Audio-visual pairings in the optic tectum are learned through early experience. Studies have shown that exposing juvenile barn owls to prismatic spectacles that displace the visual field horizontally (most commonly 23°) results in a learned displacement in the tuning curves of the corresponding auditory neurons over a number of months (Knudsen, 1985). Learned changes primarily occur in the efferent connections of the external nucleus of the inferior colliculus (ICx): an earlier auditory processing step in the auditory localization pathway that directly connects to the optic tectum (Brainard & Knudsen, 1993). These changes occur on the basis of an error signal projected back from the corrective visual input in the deeper layers of the optic tectum (Peña & DeBello, 2010). It is this re-tuning phenomenon of the auditory ICx neurons on the basis of visual input in the deeper layers of the optic tectum that the proposed model will capture.

Computational Background

We use the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2004). The NEF is used to represent, transform, and add dynamics to vectors of numbers via populations of spiking neurons, their synapses, and recurrence connections in the network.

NEF representations are an n-dimensional extension of the population coding work of Georgopoulos, Schwartz, and Kettner (1986). Neurons in the population are described in terms of an encoder and decoder that translate neural activity (a filtered spike train) to and from the vector space. The activity of a neuron can be expressed as:

$$a = G[\alpha \mathbf{e} \cdot \mathbf{x} + J_{bias}], \quad (1)$$

where G is the activation function, α is a scaling factor, $\mathbf{e}$

is the encoder, $\mathbf{x}$ is the vector to be encoded, and J_{bias} is the background current. The decoded estimate $\hat{\mathbf{x}}$ is described by the following equation:

$$\hat{\mathbf{x}}(t) = \sum_i \mathbf{d}_i a_i(t), \tag{2}$$

where $\mathbf{d}_i$ is the decoder and a_i is the activity of neuron i. Neural activity overall is computed as a spike train:

$$a_i(t) = \sum_s H(e^{-(t-t_s)/\tau_{PSC}}), \tag{3}$$

where H is the Heaviside step function, s is the set of all spikes occurring before the current time, and τ_{PSC} is the post-synaptic time constant for the connection (a neural property). The decoders are computed through a least-squares minimization of the difference between the decoded estimate and the encoded vector:

$$\begin{aligned} \mathbf{d} &= \Upsilon^{-1}\Gamma \\ \Gamma_{ij} &= \int a_i a_j dx \\ \Upsilon &= \int a_j \mathbf{x} dx \end{aligned} \tag{4}$$

We use the Prescribed Error Sensitivity (PES) supervised learning rule described in MacNeil and Eliasmith (2011) that performs the described least-squares minimization online.

$$\Delta\omega_{ij} = \kappa\alpha_j \mathbf{e}_j \cdot \mathbf{E} a_i, \tag{5}$$

where ω_{ij} is the connection weights between the ith and jth neurons, κ is a scalar learning rate, and $\mathbf{E}$ is the error vector that will be minimized. The other symbols are consistent with the previous formulas. This rule is analogous to classic perceptron delta rule with the exception that only a portion of the error signal that each neuron is sensitive to is computed for a given neuron.

The Model

We use a highly simplified model of the auditory input, visual input, ICx, and optic tectum in order to capture supervised learning in the auditory localization pathway of the barn owl. Both the auditory and visual input are described in terms of a 180° arc on the unit circle in the horizontal plane from -1.0 to 1.0.[1] This arc was represented by two values in the network corresponding to the x and y values. Though the input could vary fully over this continuous space, assessment occurred in 4° increments as we describe later.

Three ensembles of neurons describe the basic model: the ICx, the shallow optic tectum, and the deep optic tectum. Each ensemble was comprised of 400 leaky-integrate-and-fire (LIF) neurons. This neuron model is widely used for its flexibility as an approximation of a broad range of neuron models (Koch, 2004). It also functions as a limiting case of more complex models like the Hodgkin-Huxley model (Partridge, 1966). To set the α and J_{bias} parameters (Equation 1) we randomly chose these values such that the resulting ideal tuning curves generated by the LIF neuron model would involve neurons that fired over a range of inputs between ±15° and ±35°.

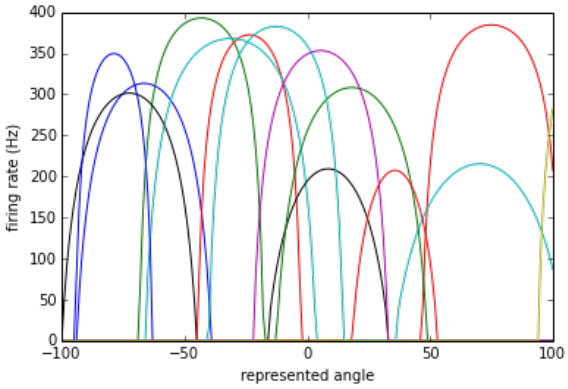


Figure 1: Ideal desired neuron tuning curves (used to generate neuron gain α and bias J_{bias} parameters).

The auditory input enters via the ICx, projects to the shallow optic tectum then to the deep optic tectum. The latter receives the visual input and projects the difference between the shallow optic tectum and the visual input as an error signal back to the ICx connections (see Fig. 2).

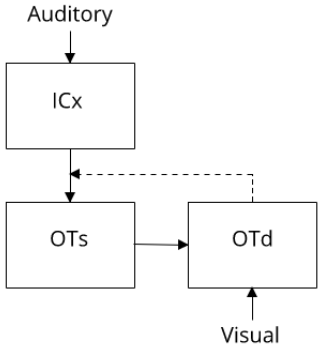


Figure 2: Basic structure of the model, where OTs is the shallow optic tectum, OTd is the deep optic tectum, and the dotted line is an error signal.

Unlike all other computational models of supervised learning in the barn owl, our model did not provide any internal structure. This means that the signal from the ICx to the shallow optic tectum originally had random weights (i.e., computed the function $f(x) = 0$). The system, therefore, had to learn the initial spatial representation prior to learning the modified representation (i.e., the introduction of the prism). It also lends a certain symmetry to the model: initial connections from the ICx to the optic tectum in the early life of the owl (i.e., before experimentation) have an identical learning

[1] This gets rotated to the left by 23° in the second training block.

procedure to subsequent learning. However, this does oversimplify some of the details in the underlying neuroscience (e.g., differences between neurotransmitters used in different learning situations).

In what follows, we describe the training and assessment procedure we used on the model.

Method

The methodological design of the study is based off of the work of Brainard and Knudsen (1998) and Linkenhoker and Knudsen (2002). We outline it, below.

The basic setup is comprised of 4 blocks: two training and two assessment. They occurred in alternating order with one of the training blocks first. Both training blocks lasted 200s (3.33min) of simulation time during which randomly selected stimuli along the 180° arc of the unit circle were presented for 500ms intervals. This resulted in 400 random stimuli being presented during each training block.

The assessment blocks determined the average activity of each neuron in 4° increments along the 180° arc of the unit circle. Each stimuli is presented for 50ms resulting in a total of 2.25s of simulation time across 45 stimuli. No learning occurred during each assessment block. The tuning curves were quantified in terms of the best tuning, which is calculated as the center of the range of values that elicited greater than 50% of the maximum response. The second assessment block used the same best tuning location as its zero point in order to see the difference across assessments.

As many more neurons were examined in the model than in the neuroscience study, we assumed that neurons with the same best tuning location were proximate to one another (consistent with their tonotopic arrangement in the ICx; Knudsen, 2002). Thus, each neuron was normalized relative to the maximum activity elicited by the pool of neurons with the same best tuning location. For example, if 4 neurons had the their best tuning (as described above) 16° left of the zero point, they would all be normalized relative to the maximum activity among them.

The second training block occurred with the prism signal present and a deviation in the visual input of 23° to the left. During the second assessment block the prism was removed, but no learning occurred to stabilize the original output. This was consistent with the original study.

Results

The results of the model were as expected. The model was able to accommodate the prism with appropriate adaptations in the ICx connections. We used the average interaural time difference (ITD; the standard measurement unit in the literature) to arc angle conversion of 2.5*u*s to 1° of the unit circle (Linkenhoker & Knudsen, 2002).

Neurons in the first assessment developed tuning curves approaching a normal curve when normalized and centered to their best tuning (see Fig. 3). The mean tuning curve was an even better approximation of normality (see Fig. 4). The maximum of the mean tuning curve had a score of 0.73.

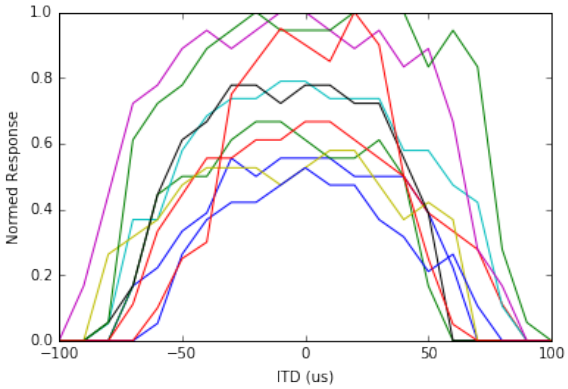


Figure 3: The learned tuning curves of 10 random neurons during the first assessment.

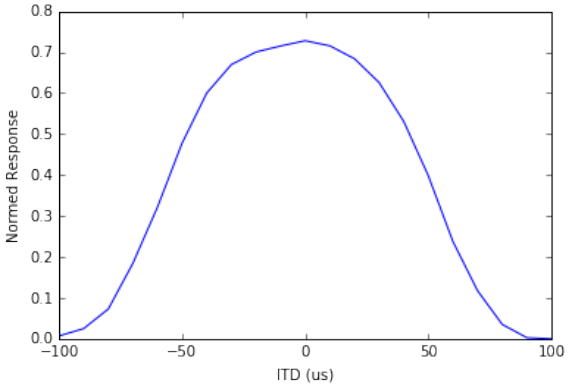


Figure 4: The mean learned tuning curve of 112 random neurons during the first assessment.

Neurons in the second assessment adapted to the prism condition by narrowing their range of activity slightly from the first training condition (see Fig. 6). The maximum activity score was 0.73. Figure 5 shows the neuron tuning curves relative to their own best tuning rather than the best tuning of the initial assessment (Fig. 7). The mean tuning curve of this population achieved a peak angle rotation of 50*u*s and 20° as a consequence of the prism (see Fig. 6).

Discussion

Recall that the goal was to model supervised learning in the localization pathway of barn owls using a simplified spiking neural network and learning rule. Given that the owls in the original empirical study had their initial training (birth until assessment) and re-training (prism adaptation) occur on the order of months instead of minutes, the results are very promising. The average tuning curve was more regular in shape than that expected by the original experiment (see Fig. 4 and 9, respectively). Nevertheless, the model's results are consistent with the corresponding empirical research on the whole.

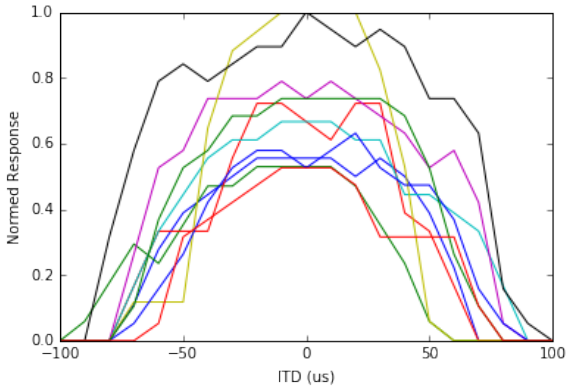


Figure 5: The learned tuning curves of 10 random neurons during the second assessment relative to their own best tuning curves (i.e., within the context of the prism).

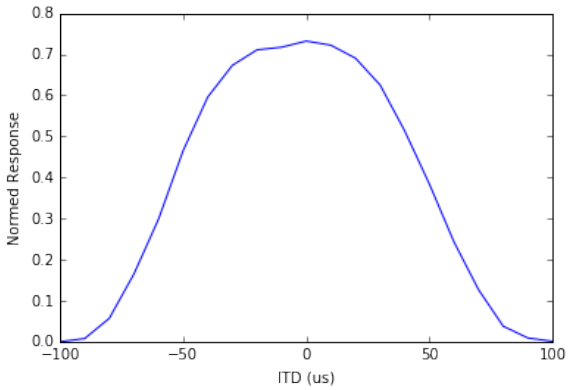


Figure 6: The learned tuning curves of 105 random neurons during the second assessment relative to their own best tuning curves (i.e., within the context of the prism).

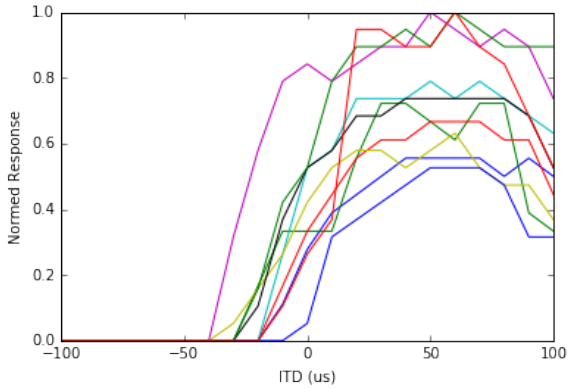


Figure 7: The learned tuning curves of 10 random neurons during the second assessment relative to the best tuning curves from the first assessment.

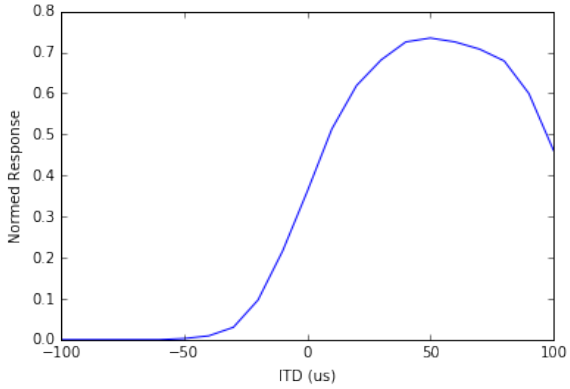


Figure 8: The learned tuning curves of the same 112 neurons from the first assessment during the second assessment relative to the best tuning curves from the first assessment.

In the original study, juvenile owls achieved a median rotation of 43*u*s (17°) with an ideal rotation of 57*u*s for a 23° prism deviation (Brainard & Knudsen, 1998; Linkenhoker & Knudsen, 2002). Our score of 50*u*s is placed in between these two values. We expect that this is a consequence of having our learning rate too high. Lower learning rates require longer time lines, which often require specialized hardware as the model's requirements increase both with the size of the neuron populations and the length of time that is simulated.

The current model is a preliminary prototype to determine whether it would be worthwhile to commit the computational effort to run this model at a much larger scale. Our results suggest that it is worth scaling up the model in future work. This would then allow us to more deeply explore effects of the learning rate as well as the learning trajectory during the prism condition.

The generalizability of this technique is much stronger than all of the other computational models. Many of the models, including Witten et al. (2008), require that the individual neurons are already sensitized to a given location and often only that location. This model learns to represent a distribution of values with a simple learning rule (PES). The system, based on its training input, manages to capture tuning curves that, on average, are very similar to what one would expect a real neuron to have. It is then able to accommodate the prism deviations from within this learned framework.

The current model uses a post-synaptic time constant in the scale of α-amino-3-hydroxy-5-methyl-4-isoxazole propionic acid (AMPA; 5ms), which is known to be a contributing factor for the learned signals (e.g., prior to the prism condition; Knudsen, 2002). Research suggests that there are at least two other neurotransmitters that contribute to supervised learning in the barn owl: *N*-methyl-*D*-aspartate (NMDA) and γ-aminobutyric acid type A ($GABA_A$). AMPA contributes to learning the new signals (during the prism condition) and $GABA_A$ regulates between the two. GABAergic inhibition also seems to have a gating-like effect on the learning process overall. It determines when the error signal is propagated

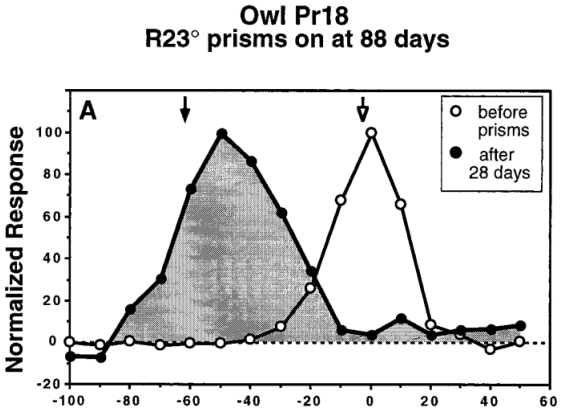


Figure 9: Original graph of the interaural time differences across the prism and no prism conditions, included with permission from Brainard and Knudsen (1998).

from the deeper layers of the optic tectum to the ICx (Peña & DeBello, 2010). We would like to begin incorporating these more detailed aspects into future instantiations of the model.

A number of additional extensions from the neuroscience literature are readily amenable to the model. The learning trajectories associated with larger time frames, in particular, show a number of interesting details. First, juvenile barn owls do not learn to adapt to the prism spectacles until after they are 60 days old, even if they are attached to them at as early as 13 days (Brainard & Knudsen, 1998). Our current research suggests that this property might be a consequence of changes in the initial tuning curves exemplified in Fig. 1. Second, after about 200 days learning tapers off such that effectively no learning occurs after about a year in the standard paradigm (Brainard & Knudsen, 1998; Linkenhoker & Knudsen, 2002). However, as a third detail, learning *is* actually possible in yearling owls and older if a special incremental learning paradigm is used (Linkenhoker & Knudsen, 2002). Adaptations in the learning parameters are particularly amenable to this situation, with a general trend towards higher tolerance learning rates over time. Using dedicated neural simulation hardware, we intend to explore these critical periods and paradigms at full-scale (i.e., day for day of simulation to real time).

Conclusion

Recall that the goal was to model a large-scale system, with minimal global topological structure, no local internal structure, and a simple biologically plausible online local learning rule that captures supervised learning in the barn owl. As a model organism for supervised learning, the barn owl lends itself to computational models of learning. We outlined in broad strokes how the deeper layers of the optic tectum motivates this learning in the neural projections from the external nucleus of the inferior colliculus (ICx) via discrepancies between auditory input and visual input. We outlined the NEF framework, and described how it can be used to map neural activity to vector representations. We also described a simple learning rule for our system (PES).

In subsequent sections, we outlined how our computational model corresponds to both the underlying neuroscience and the experimental paradigm used in the relevant prism studies on the barn owl. We showed that our model is able to capture the basic outcomes of this experimental research despite learning the initial tuning curves (the first training block) and a much more restricted time frame. We outlined some variations between our model and the neuroscience outcomes, mainly in terms of an overly strong learning rate. We then suggested some future extensions of the research in terms of larger models and time frames, more detailed analyses of the learning parameters, and more detailed model designs. Just as the barn owl is a (simple) model organism for the complexity of supervised learning in general, our model also functions as a simple but powerful computational instantiation of the complexity of the model organism.

References

Brainard, M. S., & Knudsen, E. I. (1993). Experience-dependent plasticity in the inferior colliculus: a site for visual calibration of the neural representation of auditory space in the barn owl. *The Journal of Neuroscience*, *13*(11), 4589–4608.

Brainard, M. S., & Knudsen, E. I. (1998). Sensitive periods for visual calibration of the auditory space map in the barn owl optic tectum. *The Journal of Neuroscience*, *18*(10), 3929–3942.

Brainard, M. S., Knudsen, E. I., & Esterly, S. D. (1992). Neural derivation of sound source location: resolution of spatial ambiguities in binaural cues. *The Journal of the Acoustical Society of America*, *91*(2), 1015–1027.

D'Souza, P., Liu, S.-C., & Hahnloser, R. H. (2010). Perceptron learning rule derived from spike-frequency adaptation and spike-time-dependent plasticity. *Proceedings of the National Academy of Sciences*, *107*(10), 4722–4727.

Eliasmith, C., & Anderson, C. H. (2004). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Fischer, B. J., Anderson, C. H., & Peña, J. L. (2009). Multiplicative auditory spatial receptive fields created by a hierarchy of population codes. *PloS one*, *4*(11), e8015.

Georgopoulos, A. P., Schwartz, A. B., & Kettner, R. E. (1986). Neuronal population coding of movement direction. *Science*, *233*(4771), 1416–1419.

Huo, J., & Murray, A. (2009). The adaptation of visual and auditory integration in the barn owl superior colliculus with spike timing dependent plasticity. *Neural Networks*, *22*(7), 913–921.

Huo, J., Murray, A., & Wei, D. (2012). Adaptive visual and auditory map alignment in barn owl superior colliculus and its neuromorphic implementation. *Neural Networks*

and Learning Systems, IEEE Transactions on, *23*(9), 1486–1497.

Knudsen, E. I. (1985). Experience alters the spatial tuning of auditory units in the optic tectum during a sensitive period in the barn owl. *The Journal of Neuroscience*, *5*(11), 3094–3109.

Knudsen, E. I. (1994). Supervised learning in the brain. *Journal of Neuroscience*, *14*(7), 3985–3997.

Knudsen, E. I. (2002). Instructed learning in the auditory localization pathway of the barn owl. *Nature*, *417*(6886), 322–328.

Knudsen, E. I., Blasdel, G. G., & Konishi, M. (1979). Sound localization by the barn owl (tyto alba) measured with the search coil technique. *Journal of comparative physiology*, *133*(1), 1–11.

Koch, C. (2004). *Biophysics of computation: information processing in single neurons*. Oxford university press.

Linkenhoker, B. A., & Knudsen, E. I. (2002). Incremental training increases the plasticity of the auditory space map in adult barn owls. *Nature*, *419*(6904), 293–296.

MacNeil, D., & Eliasmith, C. (2011). Fine-tuning and the stability of recurrent neural networks. *PloS one*, *6*(9), e22885.

Partridge, L. D. (1966). A possible source of nerve signal distortion arising in pulse rate encoding of signals. *Journal of theoretical biology*, *11*(2), 257–281.

Peña, J. L., & DeBello, W. M. (2010). Auditory processing, plasticity, and learning in the barn owl. *ILAR journal*, *51*(4), 338–352.

Witten, I. B., Knudsen, E. I., & Sompolinsky, H. (2008). A hebbian learning rule mediates asymmetric plasticity in aligning sensory representations. *Journal of neurophysiology*, *100*(2), 1067–1079.

This page intentionally blank.

Poster Abstracts

This page intentionally blank.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Examining load-inducing factors in instructional design: An ACT-R approach

Maria Wirzberger (maria.wirzberger@phil.tu-chemnitz.de)
E-Learning and New Media, Technische Universität Chemnitz,
Straße der Nationen 12, 09111 Chemnitz, Germany

Günter Daniel Rey (guenter-daniel.rey@phil.tu-chemnitz.de)
E-Learning and New Media, Technische Universität Chemnitz,
Straße der Nationen 12, 09111 Chemnitz, Germany

Keywords: Cognitive Load, Instructional Design, ACT-R

Introduction

In multimedia-based learning settings, limitations in mental resource capacity have to be taken into account to avoid impairing effects on learning performance. Despite the enhanced potential in capturing motivation and engagement, the multimodal, interactive and often distributed presentation of information within such settings is heavily prone to overload learners' mental facilities. To be able to handle the arising challenges, factors and effects related to the associated resource demands should be investigated in a more detailed way.

Theoretical background

A prominent and influential theory that provides versatile advice for the conducive design of media-transmitted instructional content from a cognitive perspective is the Cognitive Load Theory (Sweller, 1988; Sweller, Ayres, & Kalyuga, 2011). It is concerned with the question in what way learning scenarios demand learners' cognitive resources, since without knowing anything about underlying human cognition, instructional design is blind (Sweller et al., 2011). Amongst its basic assumptions, the theory postulates a practically unlimited storage capacity of long-term memory, the mental representation and organization of knowledge via schemata, and a limitation of working memory in terms of duration and capacity. Mental resource demands related to learning situations arise from three sources: While task complexity based on learners' previous knowledge constitutes *intrinsic load*, effects of inappropriate instructional presentation add to *extraneous load*. Both aspects affect performance on a structural and short-term level. By contrast, schema acquisition and automation, characterizing *germane load*, have to be considered on processual and long-term accounts. According to the theory, learning performance is impaired if the total amount of processing requirements exceeds the limited capacity of human working memory.

Project focus

This project focusses on the question, how load induced due to schema acquisition changes over time while working on a learning task. Besides of distinct cognitive mechanisms in different stages over the learning process, the influence of structural load facets in this context will be investigated.

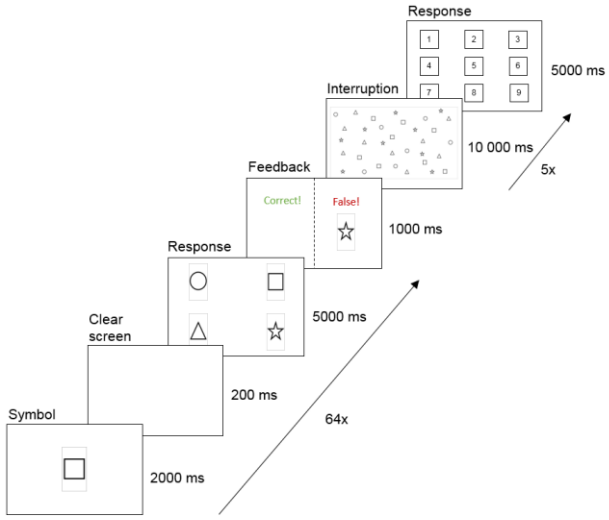


Fig. 1. Schematic trial structure. Presentation of second symbol analog and both separated by clear screen.

Task setting

In the first instance, a basic learning task is used to approach the focus of interest. Compared to comprehensive learning settings, such facilitates a more concise and controllable inspection of underlying cognitive mechanisms and processes. The chosen task (see **Fig. 1**) requires participants to figure out and memorize combinations of arbitrary geometric symbols. They are presented one or two symbols one after another and have to indicate which symbol completes the combination by selecting the correct symbol from an offered choice on the screen. For instance, a circle and a square being displayed would result in choosing a star. Such combinations have to be remembered and constitute the knowledge schema obtained over the task. The number of symbols determining the following symbol represents the intrinsic load component that is varied between subjects. An interrupting secondary task induced at defined stages during the assignment characterizes the extraneous load component that is included as within-subjects variable. Within the secondary task, participants have to search for and count instances of two selected types of geometric symbols from a picture, for example all circles

and stars, and indicate their numbers afterwards. Both structural load components are considered as independent variables in this setting. Learning performance is recorded continuously via correctness and duration of responses. The resulting efficiency score reflects the amount of mental resources invested to acquire the task-related schema (germane load component) and serves as dependent variable.

Experimental results

Preliminary results from an already conducted human experimental setting with 116 student participants (93 female, M_{age} = 23.25 years, range: 18-44 years) confirm influences of both structural load features on the observed learning performance. Apart from differing patterns of performance for easy and difficult versions of the task, they indicate a specific loss pattern in performance due to the interruptions especially in the easy condition. Based on that findings, various open questions on relevant cognitive mechanisms underlying the aspired temporal model of load progression arise.

Load measurement

In general, when attempting to investigate such issue, common approaches of load measurement by subjective questionnaires or physiological indicators face limitations in terms of diagnosticity and sensitivity. Experimentally manipulated performance measurement indeed provides a controlled way of assessment, but merely operates on indirect means as well and therefore lacks accessibility. On that point, the cognitive architecture ACT-R (Anderson & Lebiere, 1998; Anderson, 2007) offers the opportunity to clarify cognitive determinants that potentially underlie the observed performance. Implementing such a model structure raises the need to clearly think about each step relating to a given task, to ensure compatibility with founded psychological theories on human information processing.

Model development

The developed cognitive model will take into account existing work on the acquisition of complex cognitive skills (Anderson, 1982; Van Merriënboer, 1997; Taatgen & Lee, 2003). In correspondence with Bartlett (1932) and Gagné and Dick (1983), the formation of schemata will be addressed in both declarative and procedural manners, emphasizing the relevance of subsymbolic mechanisms like *activation*, *production compilation* or *reward*. Additionally, the model will base upon research on interruption and resumption during task processing (Trafton, Altmann, Brock, & Minz, 2003; Wirzberger & Russwinkel, 2015), since the disruptiveness of an interruption at a time is influenced by the amount and accessibility of available cognitive resources. On technical accounts, a milestone will consist in establishing a direct connection between the ACT-R model and the already existing Python-based experimental task via a JSON network interface (Hope, Schoelles, & Gray, 2014). Such methodology provides the option to link the developed model to more complex and lifelike multimedia-based learning settings prospectively. In doing so, predictions and observations from the basic scenario can be validated in richer knowledge domains, as already planned within the next step.

Conclusion

Overall, this project constitutes a fine step forward in understanding cognitive processes while acquiring knowledge from media-transmitted instructional content. In doing so, it provides relevant insights into a so far rather vague defined theoretical framework, and additionally contributes to interconnect approaches from different fields of research.

Acknowledgments

The authors gratefully acknowledge funding from the German Research Foundation (DFG), GRK 1780/1.

References

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review, 89,* 369–406.

Anderson, J. R., & Lebiere, C. J. (1998). *The atomic components of thought*. New York: Psychology Press.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Bartlett, F. C. (1932). *Remembering: An experimental and social study*. Cambridge, MA, USA: Cambridge University Press.

Gagné, R. M., & Dick, W. (1983). Instructional Psychology. *Annual Review of Psychology*, *34*, 261–295.

Hope, R. M., Schoelles, M. J., & Gray, W. D. (2014). Simplifying the interaction between cognitive models and task environments with the JSON Network Interface. *Behavior Research Methods, 46*, 1007-1012.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science, 12*, 257-285.

Sweller, J., Ayres, P., & Kalyuga, S. (2011). *Cognitive load theory*. New York: Springer Science + Business Media.

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors, 45*, 61–76.

Trafton, J. G., Altmann, E. M., Brock, D. P., & Mintz, F. E. (2003). Preparing to resume an interrupted task: Effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human-Computer Studies, 58*, 583–603.

Van Merriënboer, J. J. (1997). *Training complex cognitive skills: A four-component instructional design model for technical training*. Englewood Cliffs, NJ: Educational Technology Publications, Inc.

Wirzberger, M., & Russwinkel, N. (2015). Modeling interruption and resumption in a smartphone task: An ACT-R approach. *i-com, 14*, 147-154.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

ACT-Droid: ACT-R Interacting with Android Applications

Lisa-Madeleine Dörr (lisa-madeleine.m.doerr@campus.tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems
Technische Universität Berlin

Nele Russwinkel (nele.russwinkel@tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems
Technische Universität Berlin

Sabine Prezenski (sabine.prezenski@tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems
Technische Universität Berlin

Keywords: ACT-R; Android; granularity; usability testing; modeling; mobile context; tool.

Abstract

A tool for directly connecting ACT-R with Android applications on smartphones or tablets is introduced. The advantage of this tool is that no prototyping of the application is needed. This tool is especially useful to evaluate applications according to usability by using general modeling approaches.

Motivation

The number of Smartphone applications is growing rapidly. Likewise the demand for efficient usability testing methods is increasing. Cognitive models have the potential to meet this demand. Cognitive models developed for one application can be reused for testing similar applications (Prezenski & Russwinkel, in submission). Thus, costs of intensive user testing can be reduced to a minimum.

Nevertheless, the necessity of connecting the interface to ACT-R remains a major issue. The interface has to be translated into a format the ACT-R model can interact with. A number of tools have been developed to connect ACT-R to simulations (e.g. ACT-CV: Halbrügge, 2013, Hello Java: Büttner, 2009; Agimap: Urbas et al., 2009; SIMCog-JS: Halverson, Reynolds & Blaha, 2015, JNI: Hope, Schoelles & Gray, 2014 and others).

For tasks involving interactions with an interface one solution is to develop prototypes as, for example, in CogTool (John, Prevas, Salvucci, & Koedinger, 2004). The solution of creating prototypes of apps for the cognitive model is problematic for three reasons. First, it is a time consuming process. Second, the granularity of the prototype can affect the validity of the results. And third, depending on the specific usability questions new prototypes might be necessary, e.g. for questions addressing finer granularity.

This paper introduces ACT-Droid, a tool that allows ACT-R models to directly interact with Android smartphone applications. Thus, prototyping of applications becomes obsolete and testing usability with cognitive models a realistic goal.

With ACT-Droid no artificial tools need to be developed, ACT-Droid is a further development of Hello Java (Büttner, 2009). It directly connects to the Android app, identifies buttons and other items and can interact with them.

Technical Details

The two main tasks ACT-Droid fulfills are: performing motor output of ACT-R at the app and updating the visicon of ACT-R according to the changing app screen. These functionalities are provided by the *model interface* and the *app interface*, which communicate with each other.

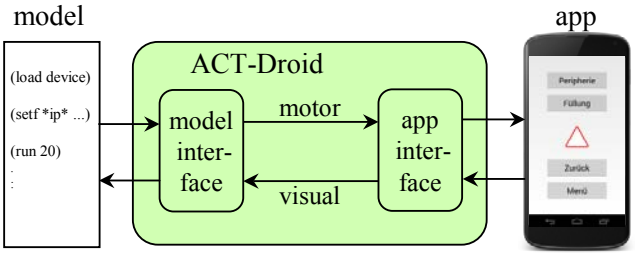


Figure 1: Architecture of ACT-Droid.

The app installed on a smartphone communicates with ACT-R over TCP/IP sockets. If the extended app is started, the *app interface* establishes a server socket and the *model interface* connects ACT-R as a client.

Motor

Currently, ACT-R's mouse commands are interpreted as fingertip touches by the Android app. So, each time the cursor is moved by the model, the *model interface* sends the new cursor position to the *app interface*. The *app interface* saves the current position of the cursor. Furthermore, if the command to click is received, the *app interface* performs a click at the saved cursor position.

Visual

The most important functionality of the *app interface* is to provide all visible information whenever the visicon of ACT-R requires updating. All visible information is recursively searched and descriptions of any visible

checkboxes, buttons and textfields are generated. This description consists of: kind, value (usually its text), color, size and position.

For the FiguRapp example (Lindner & Russwinkel, 2015) included in the figure above, the description contains four buttons (which in ACT-R is considered of the kind "oval") with their respective values "Fuellung", "Peripherie", "Zurueck" and "Menue" and a triangle of the kind "triangle" with the color "red" and no specified value.

The description provided by the *app interface* is send to the *model interface*. The *model interface* then reloads the visicon according to the received information.

What is Possible

With ACT-Droid the ACT-R model interacts with the actual Android app, the interaction is fully automated.

Furthermore, in the case of uncommon GUI elements, ACT-Droid enables the modeler to define how these elements should appear in the visicon. In the FiguRapp, for example, the image of a red triangle is defined to be of the kind "triangle" and of the color "red". But alternatively, it could also be of the kind "figure", of the color "red" and have the value "3" (for the nodes). Thus, the content of the screen can be described as detailed as necessary. The granularity is only limited by the structure of the visicon.

Due to different encodings, problems often occur with German umlauts. ACT-Droid replaces these by their respective two "normal" letters. This approach can easily be applied to other characters.

How to

The prerequisites for using ACT-Droid are the following: a computer with Lisp environment (e.g. Lispworks), ACT-R source files (the standalone does not work) and Android Studio with the source files of the app. Furthermore, an Android smartphone to run the extended app on is needed, because the emulator will not work.

ACT-Droid can be downloaded from http://dx.doi.org/10.14279/depositonce-5181. Detailed instructions in "Readme.txt" and the FiguRapp example are also included. A very basic ACT-R model that will also run on the FiguRapp is provided. This simple model will randomly explore and click on everything. To use other ACT-R models with ACT-Droid, model-interface.lisp has to be loaded and parameters have to be defined at the beginning of the model, e.g. the IP address of the smartphone.

Furthermore, a few modifications of the apps source code are necessary, i.e. adding the lispcom package and three lines of code to the apps main activity. All this is described in the material. Once the description in ACT-R's visicon is satisfying, there is no need for editing the app any further.

After everything is set up, the app on the smartphone has to be started first and then the ACT-R model can be run using the command do-experiment. The model will directly interact with the app.

Outlook

Until now, implementing scrolling with ACT-Droid has not been considered and a thorough test with different apps is pending. Another objective is the further simplification of the set-up and usage, e.g. when having more than one Android activity (common).

Currently, we are working on replacing mouse commands with the touch commands of ACT-Touch (Greene, Tamborello, & Micheals, 2013). This is the next step towards an adequate tool for efficient usability testing of apps.

References

Büttner, P. (2010). "Hello Java": Linking ACT-R 6 with a Java Simulation. In D. D. Salvucci & G. Gunzelmann (Eds.), *Proceedings of the 10th International Conference on Cognitive Modeling* (pp. 289-290). Philadelphia, PA: Drexel University.

Greene, K. K., Tamborello, F. P., & Micheals, R. J. (2013). LNCS 8007 - Computational cognitive modeling of touch and gesture on mobile multitouch devices: applications and challenges for existing theory. *Proceedings of the 15th international conference on Human-Computer Interaction* (pp. 449-455). Las Vegas, USA: ACM.

Halbrügge, M. (2013). ACT-CV: Bridging the Gap between Cognitive Models and the Outer World. In Brandenburg, E., Doria, L., Gross, A., Güntzler, T., and Smieszek, H., eds., *Proceedings of the 10th Berlin Workshop Human-Machine Systems* (pp. 205-210). Berlin, Universitätsverlag der TU Berlin.

Halverson, T. B. Reynolds, B. and Blaha., L. M. (2015) SIMCog-JS: Simplified interfacing for modeling cognition - javascript. *Proceedings of the 13th International Conference on Cognitive Modeling* (pp. 39-44). Groningen, The Netherlands.

Hope, R. M., Schoelles, M. J., & Gray, W. D. (2014). *Simplifying the interaction between cognitive models and task environments with the JSON Network Interface*. Behavior research methods, 46(4), 1007-1012.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive Human Performance Modeling Made Easy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 455–462). New York, NY, USA: ACM.

Lindner, S., & Russwinkel, N. (2015). The Effect of Design Decisions on User Expectations - A modelling approach. In C. Wienrich, T. O. Zander, & K. Gramann (Eds.), *Proceedings of the 11th Berlin Workshop Human-Machine Systems* (pp. 98–102). Berlin: Universitätsverlag der TU Berlin.

Prezenski, S., & Russwinkel, N. (in submission). Towards a general model of repeated app usage. In D. Reitter & F. E. Ritter (Eds.), *Proceedings of the 14th International Conference on Cognitive Modeling*. Pennsylvania.

Urbas, L., Heinath, M., Troesterer, S., Pape, N., Dzaack, J., Kiefer, J., & Leuchter, S. (2006). Agimap: A tool chain to support the modelling of the interaction level of dynamic

systems. *Tutorial at Trieste: Proceedings of the Seventh International Conference on Cognitive Modeling* (pp. 409). Trieste, Italy.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

The Minimalist Interference Model of the Implicit Association Test Predicts Working Memory Confounds

Michael Paul McDonald (mpmcd@uw.edu)
Department of Psychology, Guthrie Hall, 119A
Seattle, WA 98195 USA
Andrea Stocco (stocco@uw.edu)
Department of Psychology, Guthrie Hall, 119A
Seattle, WA 98195 USA
University of Washington

Keywords: interference, implicit, IAT, ACT-R, response competition

Introduction

The Implicit Association Test (IAT; Greenwald, McGhee, & Schwartz, 1998) is an indirect measure of association between concepts (e.g. race) and attributes (e.g. pleasant/unpleasant). Subjects classify concepts by category and attributes by valence as rapidly and accurately as possible, using the same response keys for concepts and attributes. Typically one key pairing is easier accomplished than with the other. It is assumed that this facility is due to an association in the subject's mind between the concepts and attributes. For instance, subjects who perform more rapidly using the white/pleasant and black/unpleasant key mapping on a Race IAT are assumed to have a positive association with White and/or a negative association with Black. This is termed an *implicit preference* for White over Black.

However, research employing the IAT has dramatically outpaced research *on* the IAT. The method yields scores with favorable psychometric properties (Cunningham, Preacher, & Banaji, 2001; Greenwald, Nosek, & Banaji, 2003), and analyses have demonstrated predictive validity by correlating IAT scores with behavioral outcomes (Greenwald, Poehlman, Uhlmann, & Banaji, 2009; Greenwald, Banaji, & Nosek, 2015). However, fundamental questions remain about the mechanism of effect, and the degree to which scores on the IAT represent underlying associations or attitudes. Understanding the mechanism of effect will allow us to better interpret D scores generated by the IAT.

Assumptions and Limitations

The end result of the IAT is the *D score*, constructed to reflect an indirect measurement of relative association strength between the target concepts and attributes. The scores are assigned positive or negative signs to indicate the direction of association. The magnitude indicates strength of effect. In the case of a Race IAT, a D near 0 would suggest neutrality, while a substantially positive D score would indicate implicit White preference, and a substantially negative score would indicate implicit Black preference.

This interpretation rests on the assumption that the IAT effect is enabled, or at least predominately contributed to, by underlying relative associations between the concepts and attributes in the subjects' minds. That is, for a subject to have a highly positive D score on the Race IAT, they must have a greater association of the concept "White" with positive than "Black" with positive, or an equivalent negative pairing.

The IAT is perhaps the most widely completed cognitive task ever developed. Through the Project Implicit website, tens of thousands of IATs are conducted each month. Complete experimental data for millions of IATs may be used for comparison to computational model results.

Presented here is a minimalist computational model of the IAT, using associations within declarative memory as the primary source of response interference.

The Minimalist Interference Model of the IAT

The minimalist interference model of the IAT (MIMI) generates an IAT effect by constructing associative interference between chunks in declarative memory. Each stimulus chunk is directly associated with its category, and the concept and attribute categories are also associated (see Figure 1). Spreading activation differentially primes the associated attribute when classifying target stimuli, thus making retrieval of the correct category more difficult. The IAT effect produced is a function of retrieval interference, and the extremity of the resulting D score is monotonic with the magnitude of disparity between underlying associations.

The effect produced yields mean latencies, but lacking SDs and realistic human noise, a D score cannot be produced (the D score, modeled after Cohen's d, is a ratio of the mean differences to the pooled standard deviation). Response time in the compatible condition is 771ms, and 819ms in the incompatible condition, with the overall mean response time at 795ms - comparable to the mean response time observed in

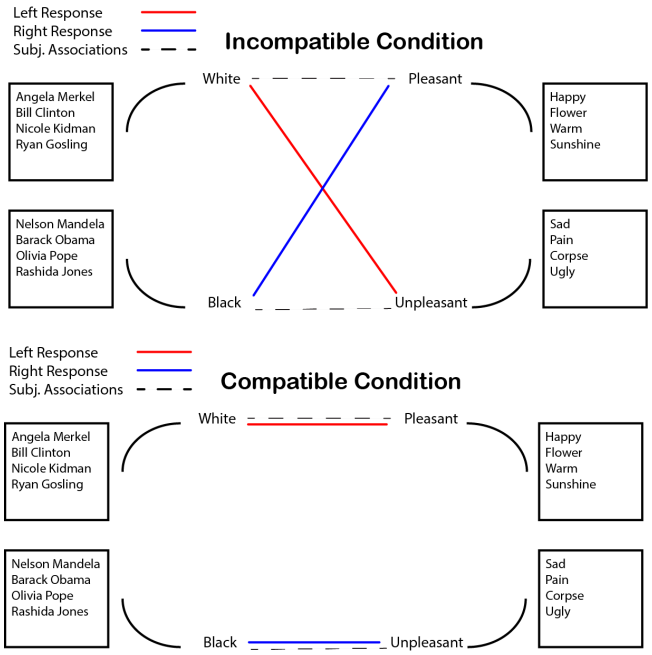


Figure 1. Diagram of associations between chunks, categories, attributes, and rules. In the compatible condition, the association created by the rules is congruent with the underlying associations in the subject's mind. In the incompatible condition, the associations created by the rules are incongruent with those in the subject's mind. This association facilitates retrieval of the rule category in the compatible condition, and detracts from performance in the incompatible condition.

the lab, about 790ms (Greenwald et al., 2003).

The modeled effect relies specifically on the underlying associations between the concepts and attributes (e.g. White and pleasant). Associations between stimuli and their parent classes are flat. The IAT effect has been shown to depend on the association between the concept category and the attributes, rather than between the exemplars and attributes (De Houwer, 2001). For example, in subjects that display automatic White preference, this preference is still evident when using uniformly negative White stimuli (e.g. Ted Bundy) and positive Black stimuli (e.g. Nelson Mandela).

When the model perceives a stimulus, it retrieves a rule with a property in common with the properties of the stimulus. This retrieval can be complicated by lingering activation from previous retrievals. For instance, if a White stimulus appears after classifying a pleasant word, the White-pleasant association may trigger a retrieval of an incorrect rule (e.g. Black-pleasant > respond with right key). Even if this doesn't cause an incorrect retrieval, the closer levels of activation make the retrieval more difficult (i.e. take longer) than in the compatible condition.

Discussion

The minimalist model approximates normal subject performance, but does not adequately explain the interference effect. MIMI assumes that the underlying strategy used by subjects is unchanged, and that differential latency between conditions is a result only of retrieval interference. While this model provides useful insight into the role of retrieval interference caused by spreading activation, it is not expected that this model accurately reflects reality. For instance, this model does not reproduce more extreme D scores in subjects that experience the compatible condition first (seen in human subjects), as the interference produced is be the same regardless of order of experience (since no learning is involved).

It is hoped that these models will lead to a more cogent understanding of the underpinning mechanisms of the IAT and other implicit interference effects. This understanding will in turn give greater insight into the proper interpretation of metrics such at the IAT's D score.

References

Cunningham, W. A., Preacher, K. J., & Banaji, M. R. (2001, March). Implicit Attitude Measures: Consistency, Stability, and Convergent Validity. *Psychological Science*, *12*(2), 163–170.

De Houwer, J. (2001, November). A Structural and Process Analysis of the Implicit Association Test. *Journal of Experimental Social Psychology*, *37*(6), 443–451.

Greenwald, A. G., Banaji, M. R., & Nosek, B. A. (2015). Statistically small effects of the Implicit Association Test can have societally large effects. *Journal of Personality and Social Psychology*, *108*(4), 553–561.

Greenwald, A. G., McGhee, D. E., & Schwartz, J. L. K. (1998). Measuring individual differences in implicit cognition: The implicit association test. *Journal of Personality and Social Psychology*, *74*(6), 1464–1480.

Greenwald, A. G., Nosek, B. A., & Banaji, M. R. (2003). Understanding and using the Implicit Association Test: I. An improved scoring algorithm. *Journal of Personality and Social Psychology*, *85*(2), 197–216.

Greenwald, A. G., Poehlman, T. A., Uhlmann, E. L., & Banaji, M. R. (2009). Understanding and using the Implicit Association Test: III. Meta-analysis of predictive validity. *Journal of Personality and Social Psychology*, *97*(1), 17–41.

Mierke, J., & Klauer, K. C. (2003). Method-Specific Variance in the Implicit Association Test. *Journal of Personality and Social Psychology*, *85*(6), 1180–1192.

von Stülpnagel, R., & Steffens, M. C. (2010). Prejudiced or Just Smart? *Zeitschrift für Psychologie / Journal of Psychology*, *218*(1), 51–53.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Distinguishing Cognitive Models of Spatial Language Understanding

Thomas Kluth (tkluth@cit-ec.uni-bielefeld.de)
Michele Burigo (mburigo@cit-ec.uni-bielefeld.de)
CITEC (Cognitive Interaction Technology Excellence Cluster), Bielefeld University, Inspiration 1, 33619 Bielefeld, Germany

Holger Schultheis (schulth@informatik.uni-bremen.de)
Cognitive Systems Group, Department of Computer Science, University of Bremen, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany

Pia Knoeferle (pia.knoeferle@hu-berlin.de)
Department of German Language and Linguistics, Humboldt-University Berlin, Unter den Linden 6, 10099 Berlin

Keywords: spatial language; visual attention; model predictions; model flexibility.

Introduction

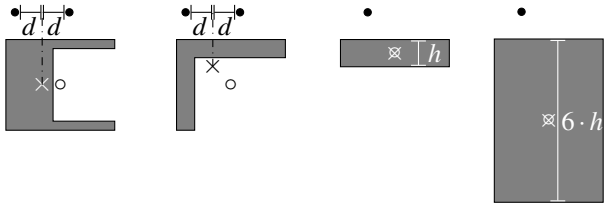
Consider the following sentence: "The picture (located object, LO) is above the desk (reference object, RO)." Given the locations of the picture and the desk – how acceptable is the use of "above"? Regier and Carlson (2001) proposed a cognitive model that computes an acceptability rating for the spatial preposition "above" in describing the spatial relation between a RO and a LO: the Attentional Vector Sum (AVS) model. In line with Logan and Sadler (1996), the AVS model assumes a shift of attention from the RO to the LO. However, in a study by Burigo and Knoeferle (2015) overt gaze shifts from the RO to the LO were infrequent during the comprehension of spatial relation utterances. By contrast, shifts in line with the mention of objects (from the LO to the RO) were highly frequent, suggesting they may be sufficient for understanding a spatial description (see also Roth & Franconeri, 2012).

Accordingly, Kluth, Burigo, and Knoeferle (2016) propose the reversed AVS (rAVS) model in which attention shifts from the LO to the RO (instead of shifting from the RO to the LO). The rAVS model accounts as well as the AVS model for the empirical data from Regier and Carlson (2001; see Kluth et al., 2016, for details). Thus, using these already existing data the two models cannot be distinguished, despite their different implementation of the attentional shift.

In order to assess whether one of these two models reflects human ratings of spatial language better than the other, we designed stimuli for which we hypothesized the models predict different acceptability ratings (see Fig. 1). With these stimuli, we conducted an empirical study to test the predictions.

Model Predictions

Based on the mechanisms of the models, we hypothesized two types of predictions for the stimuli in Fig. 1. The first type concerns the influence of asymmetrical ROs on the acceptability of spatial prepositions. Consider two LOs with equal horizontal distance d from the center-of-mass of an asymmetrical RO, as shown in Fig. 1a. The rAVS model predicts no difference in ratings for these LOs, because its computation is based on the center-of-mass of the RO. The AVS model, however, seems to predict higher ratings for the LO above the mass of the RO compared to the LO above the cavity of the RO. This is because the AVS model defines its population of vectors based on all points of the RO and thus gives more importance to the mass of the RO. The same reasoning applies for the LOs above the asymmetrical RO in Fig. 1b.



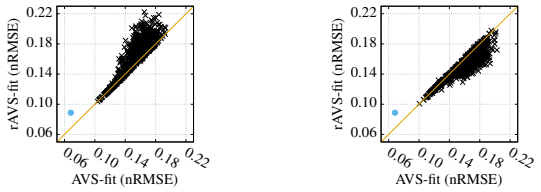
(a) Asym. "C" (b) Asym. "L" (c) "Thin" rect. (d) "Tall" rect.

Figure 1: Stimuli used for the computational and empirical studies. (• = LO; × = center-of-mass, ○ = center-of-object).

For the second type of predictions consider the two rectangular ROs in Figs. 1c and 1d. Here, the rAVS model predicts a lower rating for the LO above the "thin" rectangle compared to the LO above the "tall" rectangle. This is because the rAVS model explicity uses the *relative* distance of an LO from an RO. Here, relative distance is defined as absolute distance divided by the dimensions of the RO. Due to the free parameters of the AVS model, the prediction of the AVS model for this condition is unclear.

Empirical Study We were interested to see whether humans follow these hypothesized predictions. Thus, we conducted an empirical rating study with 28 LOs above each of the ROs in Fig. 1. Participants had to rate how well the German sentence "Der Punkt ist über dem Objekt." ("The dot is above the object.") describes a depicted RO-LO configuration. We also tested "unter" ("below") but do not report the results here. For the relative distance condition, we found that LOs above the "tall" rectangle were rated higher than LOs above the "thin" rectangle (mean difference: 0.078; 95% confidence intervals: 0.151, 0.007). This is in line with the prediction of the rAVS model.

For the asymmetrical ROs, however, we found an effect that falsifies both models: LOs above the mass of an RO were rated lower than LOs above the cavity of an RO (mean difference: 0.518; 95% confidence intervals: 0.619, 0.428). This effect contradicts the influence of the center-of-mass orientation as suggested by Regier and Carlson (2001). Neither the AVS model nor the rAVS model can account for this empiri-



(a) AVS generated data (b) rAVS generated data

Figure 2: Results of the landscaping method (×) and fits of the models to empirical data (●).

cal finding, although both models account successfully for the data from Regier and Carlson (2001; see Kluth et al., 2016). In order to better understand the performance of both models on our stimuli, we analyzed the outcomes of the models for these stimuli using three different methods.

Parameter Space Partitioning Analysis To verify whether the two models actually generate our hypothesized predictions, we applied the PSP algorithm proposed by Pitt, Kim, Navarro, and Myung (2006; using their MATLAB implementation) with the ROs shown in Fig. 1 and up to 28 LOs above each RO. This analysis confirmed the hypothesized predictions for the rAVS model but disconfirmed the hypothesized predictions for the AVS model: The AVS model is able to generate the same patterns as the rAVS model but interestingly none of the patterns hypothesized above. Arguably then, the mechanisms of the AVS model are harder to translate into testable predictions. Moreover, the AVS model generates a greater range of possible outcomes, i.e., it is more flexible than the rAVS model. However, neither model generates the empirical pattern for the asymmetrical ROs.

Model Mimicry The PSP analysis revealed that both models are able to generate the same qualitative patterns for the stimuli in Fig. 1. To investigate the relative performance of the models on these stimuli, we used the landscaping analysis as proposed by Navarro, Pitt, and Myung (2004). Apart from assessing the ability of each model to mimic the other, this method also gives us another measure of model flexibility. We generated 1000 data sets from each model and fitted both models on these artificial data by minimizing the normalized Root Mean Square Error (nRMSE). The results are shown in Fig. 2. Model fits to the empirical data (nRMSE) are plotted as filled circles. There is a slight trend that the rAVS model fits the data generated by the AVS model worse than the AVS model fits the data generated by the rAVS model (maximal rAVS fit in Fig. 2a is greater than maximal AVS fit in Fig. 2b). However, overall, both models fit their own data better than the other data and thus, no model mimics the other model.

Model Flexibility Analysis The AVS model showed greater flexibility in the PSP analysis but not in the landscaping method. The MFA proposed by Veksler, Myers, and Gluck (2015) provides a quantitative measure of model flexibility (see Veksler et al., 2015, for the relation of the MFA to PSP and landscaping). We computed the MFA for the stimuli in Fig. 1 as well as for the stimuli used by Regier and Carlson (2001). We split the range of each of the four free parameters in 50 intervals and followed the procedure outlined by Veksler et al. (2015). As indicated by the lower ϕ values in Table 1, the rAVS model is less flexible than the AVS model.

Table 1: ϕ values of the Model Flexibility Analysis (MFA). The lower the ϕ value, the less flexible the model.

	stimuli from Fig. 1	stimuli from Regier and Carlson (2001)
AVS	$\phi = 0.000899$	$\phi = 0.000420$
rAVS	$\phi = 0.000544$	$\phi = 0.000292$

Future Work In contrast to the landscaping results, the PSP and the MFA suggest that the AVS model is more flexible than the rAVS model. At the moment, we are investigating the cause of these differences in the model results.

More importantly, some of our empirical findings corroborate the rAVS model (effect of relative distance) while other findings falsify both models (effect of asymmetrical ROs). Currently, we are developing slightly modified models incorporating our suggestion that people base their acceptability ratings on the center-of-object (see ○ in Fig. 1) instead of on the center-of-mass (as suggested by Regier & Carlson, 2001). Preliminary simulation results support the use of the center-of-object over the center-of-mass.

References

Burigo, M., & Knoeferle, P. (2015). Visual attention during spatial language comprehension. *PLoS ONE*, *10*(1), e0115758.

Kluth, T., Burigo, M., & Knoeferle, P. (2016). Shifts of attention during spatial language comprehension: A computational investigation. In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence – Volume 2* (pp. 213–222). SCITEPRESS.

Logan, G. D., & Sadler, D. D. (1996). A computational analysis of the apprehension of spatial relations. In P. Bloom, M. A. Peterson, L. Nadel, & M. F. Garrett (Eds.), *Language and Space* (pp. 493–530). The MIT Press.

Navarro, D. J., Pitt, M. A., & Myung, I. J. (2004). Assessing the distinguishability of models and the informativeness of data. *Cognitive Psychology*, *49*(1), 47–84.

Pitt, M. A., Kim, W., Navarro, D. J., & Myung, J. I. (2006). Global model analysis by parameter space partitioning. *Psychological Review*, *113*(1), 57–83.

Regier, T., & Carlson, L. A. (2001). Grounding spatial language in perception: An empirical and computational investigation. *Journal of Experimental Psychology: General*, *130*(2), 273–298.

Roth, J. C., & Franconeri, S. L. (2012). Asymmetric coding of categorical spatial relations in both language and vision. *Frontiers in Psychology*, *3*(464).

Veksler, V. D., Myers, C. W., & Gluck, K. A. (2015). Model flexibility analysis. *Psychological Review*, *122*(4), 755–769.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Towards Error Recovery Microstrategies in a Touch Screen Environment

Prairie Rose Goodwin, Robert St. Amant, Rohit Aurora
prgoodwi@ncsu.edu | stamant@ncsu.edu | rarora4@ncsu.edu
Department of Computer Science, Raleigh, NC 27606, USA

Abstract

Touch screens have seen widespread adoption in the last decade due to the rise of smartphones, tablets, and touch screen laptops. While interface designs for this new interaction paradigm have improved, errors cannot be eliminated. Using an unmodified tablet and an infrared eye tracker, this paper identifies microstrategies that occur naturally during error recovery and evaluates their occurrences in low, medium, and high error environments. Cognitive modelers interested in touch screen interactions can use this information to better simulate real human performance.

Keywords: microstrategies; strategies; error; touch;

Introduction and Related Work

Current Computer Science research in touch screen errors can be broadly categorized into two areas: guideline solutions and auxiliary solutions. Guideline solutions focus on using a design that minimizes the occurrence of errors (Ng, Brewster, & Williamson, 2014) while auxiliary solutions compensate for an error after it occurs (Rudchenko, Paek, & Badger, 2011). Psychologists have used fMRIs to see what parts of the brain are involved in error recovery (Garavan, Ross, Murphy, Roche, & Stein, 2002), but to our knowledge, there is no research on the behavioral steps a person takes to recover from common errors.

Microstrategies are the low-level processes that describe the interactive behavior between the design of the available artifacts, and the cognitive, perceptual and motor processors (Gray & Boehm-Davis, 2000). When multiple strategies can be employed, the space of microstrategies can be explored to find the best explanation of human performance (Zhang & Hornof, 2014). This paper enumerates microstrategies observed during error recovery.

Microstrategies are well suited to be modeled in CPM-GOMS, because it has a straight forward notation that allows the model to be implemented more easily than it would be in a cognitive architecture like ACT-R, EPIC, or SOAR. We used a performance calculator called Cogulator (Corporation, 2014) for preliminary modeling. Like SANLab-CM (Patton & Gray, 2010) or Apex (John, Vera, Matessa, Freed, & Remington, 2002), Cogulator creates CPM-GOMS models to estimate performance.

Two new technology trends have converged in the past few years: mobile touch devices in wide use, and eye tracking technology becoming smaller and inexpensive. Modeling researchers have recently begun to explore this intersection, building and validating models of users as they carry out tasks on mobile devices. We set forth an experimental apparatus that can identify areas of interest within one degree of visual angle during bi-manual touch interactions on an unmodified tablet and an infrared eye tracker.

Methodology

Task Ten men and three women between the ages of 16 and 34 were recruited by word of mouth or volunteered at an open house. No compensation was offered to participants. Three trials were thrown out (all male) because the eye tracker's calibration did not last through the entire experiment. All glasses or contacts were removed before beginning.

Participants were told that this experiment focused on how users perceive error on a touch screen device. When the program started, a start button appeared. Once pressed, they would see five red targets to select with touch input. When a target was successfully selected, it turned grey. When all targets were selected, the screen cleared itself. After each screen, participants were asked to guess the total error rate for that screen. When a screen had been successfully cleared, ten buttons appeared with decile percentages. Participants selected the button corresponding to what they felt was the closest value. The start button then reappeared, and the process repeated for a total of 21 screens.

In this task, there were two possible kinds of errors: real target selection errors, and introduced errors. During the experiment, the system manipulated the perceived error rate. For example, if the system introduced 10% error, 1 out of 10 successful touches was ignored, simulating a missed touch.

No participant made a real target selection error. This could be because we disclosed that we were studying errors in the instructions which made people more careful, or because targets were large and spread out. Thus, all errors discussed are artificially introduced.

Experimental Apparatus Our experimental apparatus consisted of the EyeTribe eye tracker attached to the bottom-front of the Surface Pro 2 tablet using the EyeTribes proprietary harness and connected with a short cord in the USB on the left side. Both devices were attached to an adjustable height tripod with a universal 10" tablet tripod mount. The neck of the tripod was adjusted to the participant's height as needed. Participants held their hands on the side of the tablet and used their thumbs and index-finger for interactions.

This experiment followed two pilot studies that reduced noise in the eye tracking data. Like all the other eye trackers in this price range, the EyeTribe eye tracker must be placed below the screen to avoid eye lash occlusion. Some webcam-based solutions also require the tablet to be flipped so that the camera is below the screen (Wood & Bulling, 2014). Our setup specifically minimized hand occlusion during interaction. Following the examples of related work (Holland & Komogortsev, 2012), the setup was fixed on a tripod.

Microstrategies

Event logs were graphed on a timeline and patterns were found first by manual review and later parsing the log files. We divided our task into three distinct stages. We found three reoccurring microstrategies in each stage as described below.

Stage 1. Searching for target

Visual-Search (VS) Consists of multiple fixations in a row with no touch input. This microstrategy is indicative of some cognitive decision making about what action to perform next. VS occurs in 50% of all screens, and the number of fixations in the search versus its occurrence decreases in a logarithmic pattern. It is not correlated with error rate.

No-Visual-Search (NVS) Defined as the absence of *VS*.

Peripheral-Focus (PF) Consists of a single, unmoving fixation in the center of the screen during multiple touch-events. Unlike the other microstrategies, this is likely a conscious strategy by the user to keep their eyes still and only use their peripheral vision. Seen in "twitch" gaming, the user is trying to minimize reaction time by eliminating eye movement. It was only used by one participant three times, all in low-error situations. The *PF* microstrategy in Stage 2 is a continuation of this one.

Stage 2. Shifting attention away from target

No-Visual-Feedback (NVF) Consists of fixation-start, fixation-end, touch event. Both successful and unsuccessful touches are grouped together in this strategy, because they are cognitively the same action. The user anticipates the completion of a touch action without waiting for visual feedback on its success. It is seen most in low-error environments, used approximately 20% of the time.

With-Visual-Feedback (WVF) Consists of fixation-start, successful-touch, fixation-end, or fixation-start, unsuccessful-touch, repeat-touch, etc. This microstrategy is by far the most common one identified in all error levels. It occurs in 96% of all screens.

Peripheral-Focus (PF) Shift of attention is entirely cognitive, and the eyes do not move. Can result in either *NVF* or *WVF*.

Stage 3. Choosing next action

Success-With-Feedback (SWF) Consists of fixation-start, successful-touch, fixation-end. In this case, there is no need for error recovery, and the user will either return to Stage 1 to choose the next target or the task will end.

Delayed-Error-Recovery (DER) Consists of unsuccessful-touch, fixation-end, ..., touch on different target. This microstrategy is defined by some indication that the user has noticed the error but has chosen to move on and come back to it later. It is seen most in high-error environments, used 34% of the time.

Immediate-Error-Recovery (IER) In this microstrategy, fixations can be in many places. Therefore, it is only defined by an unsuccessful-touch followed by another touch on the same target indicating that they saw the error and attempt to fix it immediately. This strategy is used twice as often as *IER*, and is most seen in high-error environments and accounts for 66% of all error recoveries.

Discussion

This experiment was exploratory to identify microstrategies that occur naturally. Future studies will be designed to isolate a microstrategy to define what conditions make it more likely to occur and its duration under those circumstances.

Microstrategies and full models of some trials have been implemented in Cogulator. If the length of each fixation is specified, performance time is overestimated by roughly 10% for low-error environments. Variation increases with error rate and will be explored more in the future.

Acknowledgements

This work was funded by a grant from the NSF [IIS-1451172]. We would also like to thank Anthony Hornof and David Kieras for their modeling advice.

References

Corporation, M. (2014). *Cogulator.* Retrieved from `http://cogulator.io/index.html`

Garavan, H., Ross, T., Murphy, K., Roche, R., & Stein, E. (2002). Dissociable executive functions in the dynamic control of behavior: inhibition, error detection, and correction. *Neuroimage*, *17*(4), 1820–1829.

Gray, W. D., & Boehm-Davis, D. A. (2000). Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied*, *6*(4), 322.

Holland, C., & Komogortsev, O. (2012). Eye tracking on unmodified common tablets: Challenges and solutions. In *Proceedings of the symposium on eye tracking research and applications* (pp. 277–280).

John, B., Vera, A., Matessa, M., Freed, M., & Remington, R. (2002). Automating CPM-GOMS. In *Proc. CHI* (pp. 147–154).

Ng, A., Brewster, S. A., & Williamson, J. H. (2014). Investigating the effects of encumbrance on one-and two-handed interactions with mobile devices. In *Proc. SIGCHI* (pp. 1981–1990).

Patton, E. W., & Gray, W. D. (2010). SANLab-CM: A tool for incorporating stochastic operations into activity network modeling. *Behavior Research Methods*, *42*(3), 877–883.

Rudchenko, D., Paek, T., & Badger, E. (2011). Text text revolution: a game that improves text entry on mobile touchscreen keyboards. In *Pervasive computing* (pp. 206–213). Springer.

Wood, E., & Bulling, A. (2014). Eyetab: Model-based gaze estimation on unmodified tablet computers. In *Proceedings of the symposium on eye tracking research and applications* (pp. 207–210).

Zhang, Y., & Hornof, A. J. (2014). Understanding multitasking through parallelized strategy exploration and individualized cognitive modeling. In *Proc. CHI* (pp. 3885–3894).

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Two methods for search and optimising cognitive model parameters

David Peebles (d.peebles@hud.ac.uk)
Department of Behavioural and Social Sciences, University of Huddersfield
Queensgate, Huddersfield, HD1 3DH, UK

Keywords: Parameter optimisation, differential evolution, high throughput computing, HTCondor, ACT-R.

Introduction

Searching for the best set of parameter values is a key component of cognitive modelling and one in which a great deal of uncertainty lies. Parameter search can be a slow, laborious process when done by hand, particularly when a model has several interacting parameters, and can be challenging when models are non-differentiable, non-continuous, non-linear, stochastic, or have many local optima.

There a several methods for searching parameter spaces for such models. Here I present two: *differential evolution* (DE) and using a *High Throughput Computing* (HTC) environment managed by *HTCondor*. The two methods are similar in that they both explore parameter spaces by generating populations of models, but there the similarity ends. Below I describe both, explain the circumstances where choosing one may be preferable over the other, and provide an example of each using a simple ACT-R model for the reader to investigate.

Differential evolution

Differential evolution is an evolutionary strategy for real numbers that has been used and refined extensively for multidimensional numerical optimisation since it was devised in the mid 1990s (Storn & Price, 1995, 1997). The main attractions of DE are its simplicity, wide applicability, relatively few control parameters (three: NP, the population size, F, a scale factor applied to the mutation process, and Cr, a constant that regulates the crossover process), and the accuracy, convergence rate and robustness of its performance. To use DE for optimising cognitive model parameters, the model is conceptualised as an objective function of the parameters being optimised that produces a single fitness value (e.g., R^2) to be maximised.

The DE algorithm

In common with many evolutionary algorithms, DE applies repeated cycles of *mutation*, *recombination*, and *selection* on an initial, randomly generated population of vectors to create a single vector that produces the best solution to a problem. The DE process is started by creating an NP sized population of real-valued vectors of D dimensions, one dimension for each of the model parameters to be optimised. The vectors are initialised with uniformly distributed random numbers within maximum and minimum bounds set for each dimension.

To create the next population of vectors, each vector **i** in the current population is selected in sequence, designated as the *target* vector, and subjected to a competitive process. The competition involves the three mutation, recombination, and selection steps described below.

Mutation Mutation randomises the search process, but unlike many other evolutionary strategies that mutate vectors by adding Gaussian noise, DE does so by computing the weighted difference between two vectors in the current population. This ensures that differences in the scale and sensitivity of different vector parameters are taken into account and that the search space is explored equally on all dimensions.

A mutated *donor* vector is created by randomly selecting three unique vectors, **j**, **k** and **l**, which are not equal to **i**, from the population and adding the difference between **j** and **k** (scaled by the F parameter) to **l**.

Recombination Once the donor vector has been created it is crossed with the target vector to create the *trial* vector. This recombination allows successful solutions from the previous generation to be incorporated into the trial vector.

Crossover is achieved by a series of Bernoulli trials which determine for each of $D-1$ dimensions which parent will donate its value. The process is moderated by the crossover rate parameter Cr (where $0 \leq Cr \leq 1.$). For each dimension, a uniformly distributed random number, x between 0 and 1 is generated and compared to Cr. If $x \leq Cr$, the donor vector's parameter is passed on to the trial vector, otherwise the parameter comes from the target vector. To ensure that the trial vector does not emerge identical to the target vector, one dimension is selected at random to inherit its value from the donor vector.

Selection The model is then run with the parameter values from the trial vector and if the resulting fitness value is better than or equal to that of the target vector, the trial vector replaces it in the next generation, or else the target vector is retained in the next generation. This process of mutation, recombination, and selection is carried out for each vector in the current population until the next population is created and the evolutionary process continues for a user-defined number of cycles. The vector with the highest fitness is recorded for each population and the winning vector in the final population is considered the best solution to the problem.

Setting DE parameters

The performance of the DE algorithm is quite sensitive to its three control parameters and numerous attempts have been made to determine the optimal values for various problems (e.g., Pedersen, 2010; Neri & Tirronen, 2010; Gämperle, Müller, & Koumoutsakos, 2002). For example in the mutation process the F constant scales all of the vector parameters

equally and determines the size of the distance between the target and trial vectors. Effective values for F are generally regarded to fall between 0.4 and 1.0 with a good initial value being 0.5 (Das & Suganthan, 2011; Storn & Price, 1995).

The crossover rate parameter Cr affects the search process by regulating the probability that noisy random values enter the trial vector (raising Cr increases the likelihood that dimension values will come from the donor vector). Although views differ, Cr values between 0.3 and 0.9 are generally considered reasonable for the majority of functions.

Recommendations for the optimum population size, NP are generally specified as a function of the number of vector parameters, D, and also vary but typically range between 3D and 10D (e.g., Storn & Price, 1995; Gämperle et al., 2002).

Research into DE is very active and a number of variants and adaptations have been developed (Neri & Tirronen, 2010). The standard version described here is still widely used and performs well on many problems.

High throughput computing and HTCondor

While DE is useful for optimising models with relatively few parameters or short run times on a single computer, if models are large, complex, or are simulating the behaviour of many participants, then the computational requirements may be such that this option becomes impractical. In these circumstances, an alternative is to search the parameter space by running a population of models over a computer network and one relatively accessible and increasingly popular way to do this is by using *HTCondor* (https://research.cs.wisc.edu/htcondor).

HTCondor is an open source, cross-platform software system for managing and scheduling computationally intensive tasks across computer networks developed over many years at the University of Wisconsin-Madison (Litzkow, Livny, & Mutka, 1988). It can be employed on dedicated server clusters or to schedule tasks over idle desktop computers on a network and it is widely used in universities and research institutions worldwide, including CERN, Fermilab, and NASA.

Using HTCondor for exploring parameter spaces for cognitive models can be achieved by submitting multiple versions of the model, each with a different set of randomly generated parameter values, analysing the returned outputs, and then iterating. The process for doing so is relatively straightforward. All that is required is the creation of a *submit description file* which specifies details about the job such as the executable to be run and upon which platform, the model files to be loaded by the executable, the command to start the program running, and the number of times to run the program. As each program may also use the standard streams, files must be defined that will substitute for stdin, stdout and stderr.

For example, the extract below is from a submit description file for a job to run 100 instances of an ACT-R model defined in the file *paired.lisp*. It specifies that only 64-bit Windows machines in the network should be used, that the executable is ACT-R for 64-bit Windows, and that the arguments to the executable are to load the model file, run it for 20 participants, and then quit. In addition, output, error, and log files are defined that will be created for each instance of the model and specifications made that both the executable and the model file should be transferred to each machine. Finally, the last command sets the job to run 100 instances of the model.

```
requirements = (OpSys == "WINNT61" && Arch == "INTEL") ||
               (OpSys == "WINDOWS" && Arch == "INTEL") ||
               (OpSys == "WINDOWS" && Arch == "X86_64"))

executable = actr-s-64.exe
arguments = "-l 'paired.lisp' -e '(collect-data 20)' -e '(quit)'"

transfer_executable = ALWAYS
transfer_input_files = paired.lisp

output = out.stdout.$(Cluster).$(Process)
error = out.err.$(Cluster).$(Process)
log = out.clog.$(Cluster).$(Process)

queue 100
```

When all of the model instances have been run, their outputs will be available in numbered output files which can then be collected together and analysed.

Example code

To enable further investigation of these methods, code to optimise an ACT-R model of paired associate learning taken from Unit 4 of the ACT-R tutorials (available from the ACT-R website), together with full instructions is available on GitHub. The repository for differential evolution can be found at https://github.com/peebz/actr-paired-de while that for running the model on HTCondor is available at https://github.com/peebz/actr-paired-htc.

References

Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, *15*(1), 4–31.

Gämperle, R., Müller, S. D., & Koumoutsakos, P. (2002). A parameter study for differential evolution. *Advances in intelligent systems, fuzzy systems, evolutionary computation*, *10*, 293–298.

Litzkow, M. J., Livny, M., & Mutka, M. W. (1988, June). Condor—A hunter of idle workstations. In *Proceedings of the 8th international conference on distributed computing systems* (pp. 104–111).

Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review*, *33*(1-2), 61–106.

Pedersen, M. E. H. (2010). *Good parameters for differential evolution* (Tech. Rep. No. HL1002). www.hvass-labs.org: Hvass Laboratories.

Storn, R., & Price, K. (1995). *Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces* (Tech. Rep. No. TR-95-012). Berkeley, CA: ICSI Berkeley.

Storn, R., & Price, K. (1997). Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4), 341–359.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Predicting the Effects of In-Task Instruction During Multi-cue Diagnosis

Andrew Halsey[1], Christopher W. Myers[2], Kevin Gluck[2], & Jack Harris[2]
phillip.halsey.1.ctr@us.af.mil, christopher.myers.29@us.af.mil, kevin.gluck@us.af.mil, jack.harris.3@us.af.mil

[1]*Oak Ridge Institute for Science and Education at AFRL* [2]*Air Force Research Laboratory*

Keywords: cognitive modeling; ACT-R; fast and frugal trees

Introduction

In this research we examine a specific type of human-machine teaming, decision support systems (DSS; Power, 2008), to facilitate decision-making in an uncertain environment (Eom, Lee, Kim, and Somarajan, 1998). We extended a previously reported instance-based learning cognitive model (Myers, Gluck, Harris, Veksler, Mielke, and Boyd, 2015) to receive decision support from a machine learning algorithm. To date, no models have integrated instance-based learning and decision support, though both are well represented individually in the literature (e.g., Power, 2008; Thomson, Lebiere, Anderson, and Staszewski, 2015). We are interested in examining the strengths and deficits of integrating these as a predictive model of human-machine teaming in the context of a multi-cue diagnosis decision task.

Multi-cue Diagnosis Task

The multi-cue diagnosis task is a two-alternative forced choice task where a response is made based on available cues. In the current task, individuals diagnose "patients" for heart attacks according to three binary cues available each trial. Each cue is associated with a different "symptom"–the presence of which is probabilistic–and this information may help determine whether the patient should go to the Coronary Care Unit or standard Nurse's Bed (Green and Mehr, 1997; Marewski and Gigerenzer, 2012; Myers, et al., 2015). Feedback is provided based on the final decision. The next trial begins after delivery of feedback.

The learning difficulty within a particular multi-cue diagnosis environment is governed by the environment's rule consistency and the symptom base rates. Rule consistency is the probability that using the underlying rule results in a correct diagnosis. The rule for the current experiment was: if and only if cue2 is true and cue3 is true, then it is a heart attack and the correct response is Coronary Care Unit. Given the current rule consistency, choosing Coronary Care Unit in the presence of these symptoms would result in a correct response 80% of the time. The presence of a positive "symptom" associated with each cue was: cue1=0.25, cue2=0.40, and cue3=0.75.

Instance-Based Learning Theory and Model

Gonzalez, Lerch, and Lebiere (2003) proposed Instance-based Learning Theory (IBLT) as a process account of human learning during repeated decision-making. IBLT posits how humans identify, store, and retrieve information for the explicit purpose of making decisions within a dynamic, uncertain environment when performance feedback is provided (Gonzalez, et al., 2003).

IBLT has successfully accounted for human behavior in two-alternative forced-choice tasks (e.g., Gonzalez, and Dutt, 2011), classification tasks (Gagliardi, 2011), and dynamic tasks (Gluck, Stanley, Moore, Reitter, and Halbrugge, 2010; Reitter, 2010). We developed an IBLT model in ACT-R (Anderson, 2007; Thomson, et al., 2015) to make a decision based on a particular context (i.e., the presence of symptoms) and prior experience.

The model does not generate a response according to explicitly defined rules indicating the number and order of cues to check. Rather, the model generates its decision by using ACT-R's blending mechanism to blend over chunks and to determine cue encoding order and a stopping rule according a particular context. In the current paper, the IBLT model encoded decision support instruction similarly; rather than providing an explicit rule, decision support was represented as a collection of high feedback chunks that suggested a response given a particular context. Previous research has shown that the IBLT model is capable of predicting human behavior on a multi-cue diagnosis task (Myers, et al., 2015).

Model Evaluation

To simulate human-machine teaming, the IBLT model received decision support from a machine learner using a constrained version of the A* algorithm that constructed a decision rule with maximum expected reward. Three decision support types were tested across a single rule consistency: correct DSS (optimal rule), incorrect DSS (non-optimal rule), and no DSS. Each decision support condition completed 20 runs of 267 trials. Decision support was delivered at trial 60 to allow the machine learner to settle on an environmentally consistent optimal rule and to ensure the IBLT model had not reached asymptotic response accuracy performance.

The IBLT model goes through three different stages across the experiment: exploration, instruction, and exploitation. The focus in the current paper was on the exploration and exploitation phase, each with unique questions concerning model performance. During the exploration stage, we were interested in model behavior according to rule acquisition, rule adherence, and accuracy. Specifically, because the model generates a rule based on learning and experience, does the model find the underlying rule or does it generate an alternative rule? We examined optimal rule-adherence with respect to accuracy for insights into these questions.

After the machine learner delivers decision support instruction, it would be unsurprising that rule adherence and accuracy change. However, of interest is how the model behaves during the exploitation stage when the decision support rule is encoded in declarative memory. Two primary questions need to be addressed during this stage. First, to what extent does the model appropriately use the provided rule? Second, does the decision support coerce or force the model into a pattern of inflexible responding by suppressing exploratory behavior? In other words, if the environment transitioned to a new rule, would the model continue responding according to the provided decision support rule, or would the model still be able to detect environmental changes and subsequently adjust its strategy?

Model Predictions

Trial data from each run was binned to 9 blocks for ease of interpretation. Decision support instruction occurred at the beginning of block 3 (trial 60; indicated in Figure 1 as a dashed vertical line). Data from model accuracy and reward across blocks were identical and therefore we used model accuracy to examine model performance.

During the exploratory stage of blocks 1 and 2, the IBLT model began to learn and respond according to a self-generated rule that, according to the model, appeared to best explain the environment. Given accuracy and rule adherence, the rule generated by the model during these blocks was sub-optimal and as a result accuracy was unable to match the probability associated with the environment rule consistency.

After receiving decision support from the machine learner at block 3, the model was responsive in incorporating the provided instruction into its rule strategy (Figure 1). Correct DSS instruction resulted in increased model accuracy and rule adherence by following the underlying environmental rule. Thus, not only was the model making appropriate responses, but these were a result of adherence to the decision support rather than a decision strategy or rule self-generated by the model. Incorrect DSS and no DSS also responded expectedly according to the type of rule (or lack thereof) provided.

During the exploitation phase, after delivery of decision support, model behavior remains–to a degree–flexible and exploratory. For example, in the correct DSS condition, rule adherence of the model increases to nearly 100% then begins to drift lower in subsequent blocks. The reason for this behavioral variability is a result of the model forgetting the rule over time. Rule forgetting across blocks is gradual; rule adherence and accuracy both remain higher relative to the same metrics during the exploratory stage. However, some degree of forgetting behavior can be advantageous. It allows the model to continue responding dynamically according to the environment, and should an environmental change occur (i.e., rule change or change in probabilities), the model can detect these changes and adjust the response strategy accordingly.

Response time predictions corroborated documented deficits of the IBLT model (Myers, et al., 2015). Infrequently used chunks in declarative memory increased response time due to lower probability of recall. Response time decreased at block 7 because the chunks used thereafter included only frequently or recently used chunks. Based on the current results and previous findings (Myers et al., 2015), the IBLT model may not be capable of accurately accounting for human response times in a two-alternative forced choice task.

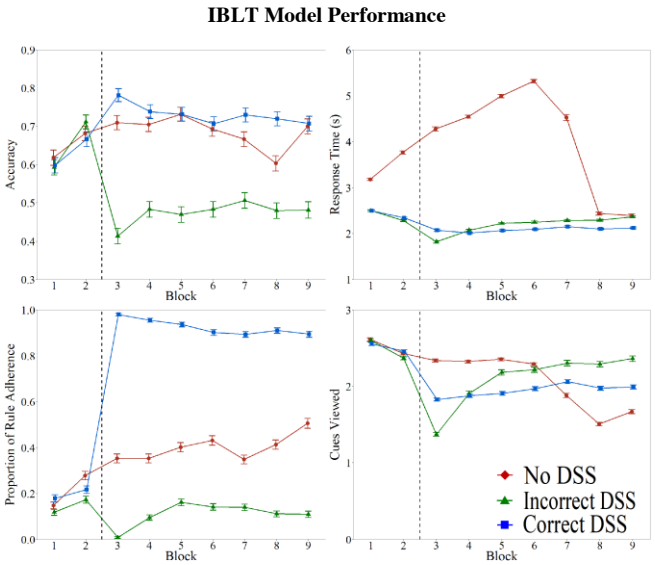


Figure 1. Performance data for the three models (+/- 1 SEM). The dashed line represents decision support.

Conclusions

The integration of the IBLT model with a machine learning decision support system demonstrated several strengths and weaknesses. The IBLT model is capable of taking instruction and incorporating it within its rule discovery strategy, as evidenced by accuracy and rule adherence changes between the exploratory and exploitation stages. The incorporation of the rule strategy does not suppress future learning. In fact, the model resumes some amount of exploratory behavior after instruction, thereby allowing the model to remain flexible and adaptive to possible environmental changes. These core strengths demonstrate a model with explanatory potential when validated against human behavior. The main weakness of the model relates to the IBLT model's inability to model response times, such as those demonstrated by humans engaged in a similar task (Myers, et al., 2005). Future research will tackle issues such as direct human to model comparisons according to instruction-taking, and determining the timing and frequency of instruction delivery.

Acknowledgments

This research was supported by the Air Force Office of Scientific Research, grant 13RH06COR.

References

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?* New York, NY: Oxford University Press.

Eom, S. B., Lee, S. M., Kim, E. B., & Somarajan, C. (1998). A survey of decision support system applications (1988-1994). *Journal of the Operational Research Society*, *49*, 109-120.

Gagliardi, F. (2011). Instance-based classifiers applied to medical databases: Diagnosis and knowledge extraction. *Artificial Intelligence in Medicine*, *52*, 123-139.

Gluck, K., Stanley, C., Moore, L., Reitter, D., & Halbrügge, M. (2010). Exploration for understanding in cognitive modeling. *Journal of Artificial General Intelligence*, *2*, 88-107.

Gonzalez, C., & Dutt, V. (2011). Instance-based learning: Integrating sampling and repeated decisions from experience. *Psychological Review*, *118*, 1-29.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, *27*, 591-635.

Green, L., & Mehr, D. R. (1997). What alters physicians' decisions to admit to the coronary care unit?. *Journal of Family Practice*, *45*, 219-226.

Kaplan, B. (2001). Evaluating informatics applications–clinical decision support systems literature review. *International Journal of Medical Informatics, 64,* 15-37.

Marewski, J. N., & Gigerenzer, G. (2012). Heuristic decision making in medicine. *Dialogues in Clinical Neuroscience*, *14*, 77-89.

Myers, C. W., Gluck, K. A., Harris, J., Veksler, V., Mielke, T., & Boyd, R. (2015). Evaluating instance-based learning in multi-cue diagnosis. In N. A. Taatgen, M. K., van Vugt, J. P Borst, & K. Mehlhorn. *Proceedings of ICCM 2015*. Paper presented at International Conference on Cognitive Science, Groningen, Netherlands (198-199).

Power, D. J. (2008). Decision support systems: A historical overview. In F. Burnstein & C. W. Holsapple (Eds.), *Handbook on decision support systems 1: Basic themes* (pp. 121-140). Springer.

Reitter, D. (2010). Metacognition and multiple strategies in a cognitive model of online control. *Journal of Artificial General Intelligence*, *2*, 20-37.

Thomson, R., Lebiere, C., Anderson, J. R., & Staszewski, J. (2015). A general instance-based learning framework for studying intuitive decision-making in a cognitive architecture. *Journal of Applied Research in Memory and Cognition, 4,* 180-190.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Eye-tracking Analysis for Product Recommendation Virtual Agent with Markov Chain Model

Tetsuya Matsui (tmatsui@nii.ac.jp)
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda Tokyo 101-8430, Japan

Seiji Yamada (seiji@nii.ac.jp)
National Institute of Informatics/Sokendai/Tokyo Institute of Technology
2-1-2 Hitotsubashi, Chiyoda, Tokyo 101-8430, Japan

Keywords: eye-tracking, product recommendation virtual agents, Markov chain, Shanon entropy, transition matrix, stationary distribution, fixation duration

Introduction

PRVAs, product recommendation virtual agents, are agents that are designed for virtual clerks in online shopping. Prendinger et al. investigated the effect of virtual clerks by eye tracking analysis (Prendinger, Ma, & Ishizuka, 2007). In their experiment, participants were introduced real estate properties by text, speech, and an animated agent. They showed that the agent's use of deictic gestures had the effect of attracting a participant's gaze. Terada et al. studied what appearance was the most suitable for PRVAs (Terada, Jing, & Yamada, 2015). They showed that one of the most effective appearances were dog, robot, and young woman. In this paper, we investigated the effect of PRVA's emotion transition to user's gaze by eye tracking analysis.

A Markov chain model is widely used for constructing a model of eye tracking transition. Liechty et al. showed local and global covert visual attention by adapting a Bayesian hidden Markov model (Liechty, Pieters, & Wedel, 2003). He et al. suggested investigating hidden user behaviors that occur when a user is using a search site by using a partially observable Markov model with duration (POMD) (He & Wang, 2011). This model is derived from the hidden Markov model (HMM). The difference was that POMD contained a partially observable event. He et al. suggested that only seeing without clicking links was the hidden user behavior.

In this paper, our goal was to improve the PRVA design methodology by analyzing user eye-tracking data. We focused on transition-based analysis. In prior research on human-agent interaction, eye-tracking data were mainly analyzed on the basis of fixation durations. This is the most important method in this paper.

Markov chain

In our research, we used the Markov chain model for analyzing the fixation transitions between areas of interest (AOI sequence). The Markov chain satisfies the following equation, where X_n is a random variable and n means time step (Brooks, Gelman, Jones, & Meng, 2011).

$$\begin{aligned}\mathbf{P}(X_{n+1} = x_{n+1} \mid X_n = x_n, \cdots, X_0 = x_0) \\ = \mathbf{P}(X_{n+1} = x_{n+1} \mid X_n = x_n)\end{aligned} \quad (1)$$

In this research, our goal was to compare the transition entropy and the stationary entropy of the AOI sequence

Experiment

Participants

Fifteen Japanese participants joined in the experiment. There was eight males and seven females, and they were aged between 20 and 39, for an average of 29.3 (SD = 6.9). Due to not getting sufficient gaze data, we omitted the data of one male participant.

Task

The PRVA recommended 10 package tours to Japanese castles. These castles were built in the Japanese Middle Ages, from about the 13th to 16th century. The PRVA made recommendations successively, and the recommendation order was random. For the first half of the recommendations, the PRVA kept a poker face without making any gestures. We defined this agent as the apathy agent. In the latter half, the PRVA smiled and made cute gestures. We defined this agent as the positive agent. This change in facial expressions and gestures expressed the agent's emotion transition, and we aimed for the agent's positive emotion to infect participants.

The PRVAs were executed with MMDAgent[1]. This is a free toolkit for constructing agent systems with speech. It contains the agent character "Mei" and is distributed by the Nagoya Institute of Technology. We also used the text to speech software VOCELOID+ Yuduki Yulari EX2[2] for the agent's voice.

Apparatus

We carried out experiments with Tobii Pro X2-60 and a 30-inch LCD monitor (1920 × 1200 resolution). Eye movements were recorded at a 60-Hz sampling rate. All participants were requested to sit down in a chair at a 60-cm distance from the monitor during the experiment. All stimuli

[1] http://www.mmdagent.jp/

[2] http://www.ah-soft.com/voiceroid/yukari/

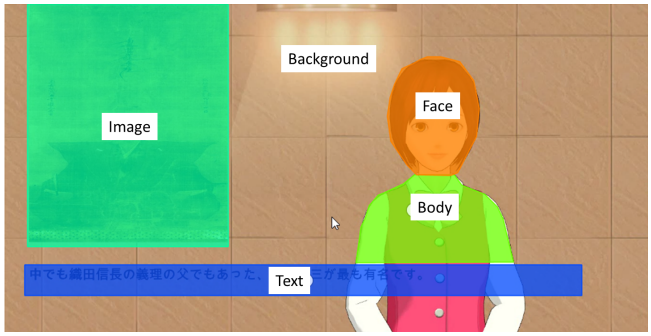


Figure 1: Defined AOIs

were presented on the monitor, and all participants listened to the recommendations with headphones. To construct a transition matrix and stationary distribution, we used the R package "markovchain" (Spedicato, Kang, & Yalamanchi, n.d.).

Analysis method

We defined the AOIs as shown in Figure 1. We divided the presented stimuli area into five areas (*"background," "body," "face," "image,"* and *"text"*). We analyzed based on the fixation order. Fixation order meant the path of a participant's fixations, and we counted the number of transitions of the AOIs that the participants fixed on (including self transitions). We constructed the transition matrix and stationary distribution from this analysis. The minimum fixation duration was 60 ms, and transition advanced one step when fixation occurred.

Results

We constructed the transition matrix and stationary distribution from the fixation order of the first half of the recommendations. We calculated each transition matrix from each recommendation. We got 10 transition matrices from one participant and got 140 transition matrices in total. We calculated the average of all matrix elements. This was for the "transition matrix derived apathy agent".

Also, we calculated the stationary distribution from this matrix. This was for the "stationary distribution derived apathy agent" ($\pi_{\mathbf{a}} = (0.22, 0.11, 0.05, 0.26, 0.37)$). On the matrix, each coordinate means these AOIs: 1 = "background," 2 = "body," 3 = "face," 4 = "image," and 5 = "text." In the stationary distribution, the same coordinate means the same AOI .

We constructed the transition matrix and stationary distribution from the latter half of the recommendations in the same way. These were for the "transition matrix derived positive agent" and "stationary distribution derived agent" ($\pi_{\mathbf{p}} = (0.22, 0.16, 0.098, 0.21, 0.31)$). The same coordinate means the same AOI in $\pi_{\mathbf{a}}$.

Discussion

From $\pi_{\mathbf{a}}$ and $\pi_{\mathbf{p}}$, we can find few definite differences. The most different element was p_3 between these two matrices. In $\pi_{\mathbf{a}}$ this means 0.05, and in $\pi_{\mathbf{p}}$, this means0.098. This coordinate means the percentage of probability that fixation transitions to "face" when the fixation is on "face" one time-step before. This shows that implementing the positive emotion caused participants' fixations to stay on the agent's face. This phenomenon proves that the participants felt more humanlikeness with the agent (Strait, Vujovic, Floerke, Scheutz, & Urry, 2015).

Conclusion

There is demand for PRVAs that have the ability to attract a user's attention to products or to themselves. This can be rephrased as the ability to attract and keep a user's fixation on the images of products or agents. We investigated the effect of implementing a positive emotion in a PRVA by analyzing eye-tracking and aimed to adapt the result to the model of designing PRVAs that attract a user's fixation. From our experiment, a positive emotion attracted participants' gaze to the agent's face. This suggest a methodology of attracting or keeping a user's gaze and buying motivations.

Acknowledgment

This study was partially supported by JSPS KAKENHI " Cognitive Interaction Design " (No.26118005).

References

Brooks, S., Gelman, A., Jones, G., & Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. CRC press.

He, Y., & Wang, K. (2011). Inferring search behaviors using partially observable Markov model with duration (POMD). In *Proceedings of WSDM'11*. Hong Kong, China.

Liechty, J., Pieters, R., & Wedel, M. (2003). Global and local covert visual attention: Evidence from a bayesian hidden markov model. *Psychometrika*, *68*(4), 519–541.

Prendinger, H., Ma, C., & Ishizuka, M. (2007). Eye movements as indices for the utility of life-like interface agents: A pilot study. *Interacting with Computers*, *19*(2), 281–292.

Spedicato, G. A., Kang, T. S., & Yalamanchi, S. B. (n.d.). The markovchain Package: A Package for Easily Handling Discrete Markov Chains in R.

Strait, M., Vujovic, L., Floerke, V., Scheutz, M., & Urry, H. (2015). Too Much Humanness for Human-Robot Interaction: Exposure to Highly Humanlike Robots Elicits Aversive Responding in Observers. In *Proceedings of CHI '15*. Seoul, Korea.

Terada, K., Jing, L., & Yamada, S. (2015). Effects of Agent Appearance on Customer Buying Motivations on Online Shopping Sites. In *Proceedings of CHI '15*. Seoul, Korea.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Automatic Generation of Analogous Problems for Written Subtraction

Christina Zeller (christina.zeller@uni-bamberg.de)
Ute Schmid (ute.schmid@uni-bamberg.de)
Cognitive Systems Group, University of Bamberg
An der Weberei 5, 96045 Bamberg, Germany

Introduction

Learning in domains such as mathematics or programming, involves the acquisition of procedural knowledge (Young & O'Shea, 1981). For example, when learning written subtraction, students need to understand and apply an algorithm for calculation of differences column by column. Erroneous solutions most often are the result of procedural bugs (Brown & Burton, 1978) such as missing or faulty rules or the application of a rule in the wrong context. If such a procedural bug is diagnosed, a strategy is needed to support the student resolving this bug. Such strategies can be: written explanations, presenting additional problems, or giving bug-related feedback such as an explanation together with a worked-out example (Narciss & Huth, 2006).

A worked-out example can be considered as an analogy to the given problem which a student could not solve correctly (Gick & Holyoak, 1983). That is, for the current (target) problem a structurally isomorphic base problem is provided where the correct solution can be demonstrated step by step. While Narciss and Huth (2006) make use of this feedback approach, they rely on predefined analogies stored together with an—also predefined—set of student problems. However, the automatic generation of such analogous problems for written subtraction can improve and facilitate feedback generation.

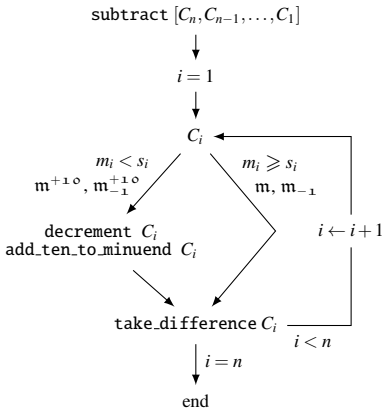
Written Subtraction

In Figure 1 the visualization of the subtraction algorithm using the *decomposition method*, which is implemented in Prolog and described in Zinn (2014), is shown.[1]

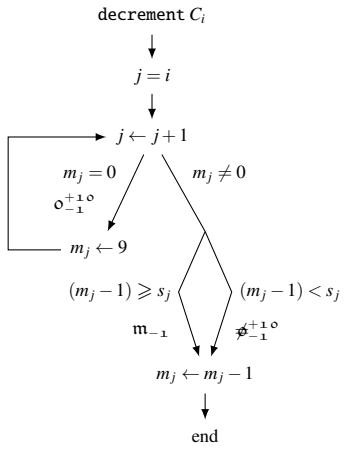
Subtraction is realized by five production rules:

- `subtract` $[C_n, C_{n-1}, ..., C_1]$: subtracts a subtrahend from a minuend. The procedure gets as input a non empty list of columns with $C_i = (m_i, s_i, d_i)$ where m_i stands for minuend, s_i for subtrahend and d_i for the difference of the column i. C_1 belongs to the rightmost and C_n to the leftmost column. If the subtrahend has fewer positions than the minuend, leading zeros are added.
- `process_column` C_i: starts with the rightmost column and compares m_i and s_i. If $m_i \leqslant s_i$ the production rule `take_difference` is applied immediately. Otherwise, a borrowing procedure is needed previously, which is the application of `decrement` and `add_ten_to_minuend`. After processing column i the next column $(i+1)$ is inspected. The `process_column` rule ends the subtraction algorithm after processing the last column ($i = n$, cf. Fig. 1a).

[1] In contrast to Zinn (2014) we label columns from right to left.



(a) Subtraction algorithm



(b) `decrement` procedure

Figure 1: Schema of the written subtraction algorithm (a) and a closer look on the `decrement` rule (b) (Zinn, 2014), enriched with the column cases ($\mathfrak{m}$, $\mathfrak{m}_{-1}$, ...; Zeller, 2015).

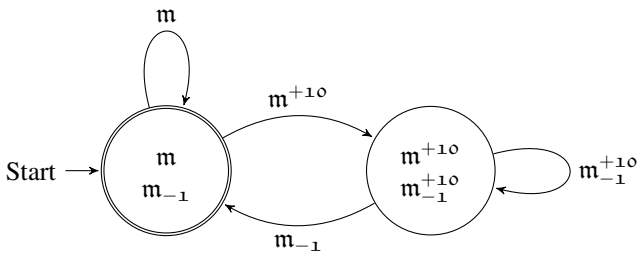


Figure 2: Automaton to describe the generation of a written subtraction problem, starting with the rightmost column using the column cases annotated in Figure 1.

- `decrement` C_i: borrows *ten* from the minuend m_{i+1}. If $m_{i+1} = 0$, further borrowing is needed in C_{i+2} (cf. Fig. 1b).
- `add_ten_to_minuend` C_i: adds the borrowed *ten* to m_i.
- `take_difference` C_i: takes the difference $m_i - s_i$ and stores the difference in d_i.

Consider the following subtraction problem:

	C_3	C_2	C_1
	3	0	5
−	2	0	6

The algorithm starts with the rightmost column (C_1). Because of $m_1 = 5$ and $s_1 = 6$ `process_column` calls the borrowing procedure. The minuend of C_2 is 0 and therefore borrowing is needed in C_3. Afterwards, `take_difference` is applied. The application of `decrement` (left), `add_ten_to_minuend` (middle), and `take_difference` (right) results in:

	2	**9**	5		2	9	**15**		2	9	15	
−	2	0	6	−	2	0	6	−	2	0	6	
												9

Next C_2 is processed (`take_difference` with $m_2 = 9$ and $s_2 = 0$). After that `take_difference` is applied to the last column (C_3). The correct difference is 99.

Analogies for Written Subtraction

The algorithm in Figure 1 induces an automaton for the generation of arbitrary subtraction problems given in Figure 2. A subtraction problem starts with the rightmost column. A column either needs borrowing (arrow $\mathfrak{m}^{+10}$) or not (arrow $\mathfrak{m}$). From the second column onward a column can be borrowed from (arrow $\mathfrak{m}_{-1}$) or be borrowed from and need borrowing simultaneously (arrow $\mathfrak{m}_{-1}^{+10}$). For this most complex case, it can be discriminated whether the value of the minuend is 0 ($\mathfrak{o}_{-1}^{+10}$) or not ($\not\mathfrak{o}_{-1}^{+10}$, cf. Fig. 1b). The states of the automaton constitute column cases, that is, they characterize the structural relation between minuend and subtrahend in each column. All subtraction problems generated with this automaton can be solved by the subtraction algorithm given in Figure 1 with the restriction that only such problems are allowed where the result is greater or equal to zero. This automaton was implemented as Prolog program (Zeller, 2015) and generated for instance the following analogous examples:

Problem 1

	C_3	C_2	C_1
	$\mathfrak{m}_{-1}$	$\mathfrak{o}_{-1}^{+10}$	$\mathfrak{m}^{+10}$
	3	0	5
−	2	0	6

Problem 2

	C_3	C_2	C_1
	$\mathfrak{m}_{-1}$	$\mathfrak{m}^{+10}$	$\mathfrak{m}$
	4	3	7
−	3	7	4

Analogy 1

	1	0	6
−	0	3	8

Analogy 2

	3	1	0
−	1	8	0

The column cases of the problem define the structure of the analogy. For example, if C_1 of the problem is of case $\mathfrak{m}^{+10}$ then this holds also for the analogy.

Conclusion

The proposed approach was integrated in an intelligent tutor system. There analogous problems were created to specifically address students' errors. That is, the analogous example preserved that characteristics of the given problem where the error occurred.

As a next step we plan an empirical study, where we want to compare automatic generated analogies with analogies created by human tutors. Here, we will start with a set of generated erroneous student solutions. These solutions will be presented to teachers in elementary schools who are experienced in teaching written subtraction. The teachers are instructed (a) to identify the error in the solution, and (b) to propose an analogues problem for which they assume that it helps the student to understand the error. Teacher solutions are analyzed with respect to the constraints of our automatic generation approach.

Furthermore, we plan to transfer the concepts to other domains. On the one hand, we are interested in transfer to related domains, such as teaching other mathematical operations (written addition, multiplication, and division). On the other hand, we are interested in transfer to other domains strongly depending on procedural skills such as teaching computer programming.

References

Brown, J. S., & Burton, R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, *2*, 155–192.

Gick, M. L., & Holyoak, K. J. (1983). Schema induction and analogical transfer. *Cognitive Psychology*, *15*, 1–38.

Narciss, S., & Huth, K. (2006). Fostering achievement and motivation with bug-related tutoring feedback in a computer based training for written subtraction. *Learning and Instruction*, *16*, 310–322.

Young, R. M., & O'Shea, T. (1981). Errors in children's subtraction. *Cognitive Science*, *5*(2), 153–177.

Zeller, C. (2015). *Automatische Erzeugung analoger Beispiele aus Debugging-Traces [Automatic generation of analogue examples from debugging-traces]* (Master's thesis, University of Bamberg, Germany). Retrieved from `http://www.cogsys.wiai.uni-bamberg.de/theses/zeller/ma_zeller-christina_online.pdf`

Zinn, C. (2014). Algorithmic debugging and literate programming to generate feedback in intelligent tutoring systems. In C. Lutz & M. Thielscher (Eds.), *KI 2014, LNCS 8736* (pp. 37–48). Springer International.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Modeling Autobiographical Memory from Photo Libraries

Junya Morita (j-morita@inf.shizuoka.ac.jp)
Faculty of Informatics, Shizuoka University. 3-5-1 Johoku, Naka-ku, Hamamatsu City, Japan

Takatsugu Hirayama (hirayama@is.nagoya-u.ac.jp), Kenji Mase (mase@nagoya-u.ac.jp)
Graduate School of Information Science, Nagoya University. Furo-cho, Chikusaku Nagoya, Japan

Kazunori Yamada (yamada.kazunori@jp.panasonic.com)
Panasonic Corporation. 1006 Kadoma, Kadoma City, Osaka, Japan

Abstract

Assuming that photographs accumulated on a personal computer reflect the life history of a person, a model of that person's autobiographical memory could be constructed. Such a model would be useful to overcome memory problems caused by factors such as aging. On the basis of this idea, we constructed a photo slideshow system comprising an ACT-R model with a private photo library.

Keywords: Photographs, Autobiographical Memory, ACT-R.

Introduction

Recalling autobiographical memory engenders a state of consciousness, called mental time travel, in which relevant memories of past events are evoked (Schacter, Addis, & Buckner, 2007; Tulving, 1985). Memory recall of a personal golden age is also said to bring psychological health and well-being (Routledge, Wildschut, Sedikides, & Juhl, 2013). On the basis of these assumptions, activities such as life reviews and reminiscences are conducted to support the elderly.

Our long-term goal is to develop a model-based method of life review and reminiscences, in which a computerized user-model guides user's mental time travel. To establish this, we developed a photo slideshow system by using ACT-R as a user-modeling platform (Anderson, Boyle, & Reiser, 1985; Anderson, 2007). In this framework, a cognitive model of a user's autobiographical memory is developed by extracting user-specific knowledge from a private photo database.

In this framework, the model and user simultaneously observe a photo retrieved by the model. When a memory recalled by the model satisfactorily fits that of the user, the photo presented by the model can generate a positive feeling in the user through synchronization effects (Chartrand & Bargh, 1999). Such synchronization effects can be strengthened by modulating parameters of the model utilizing feedbacks from users. Therefore, we assume that this slideshow can be used for not only motivating a user by presenting favored photos but also diagnosing mental states through user feedback of the presented photos.

The model

This document briefly presents the construction of the model, which uses the visual, declarative, goal and production modules of ACT-R to retrieve photos from a photo library.

Photo data and Visual Module

The outputs from a consumer-based image processing engine are used as inputs to the model. Many recent photo libraries have face detection modules. They can also recognize personal names through human-in-the-loop training. We used these functions implemented in iPhoto of Mac OSX. We also used ReKognition API (https://rekognition.com) to analyze the scenes in photos. ReKognition API is an image recognition engine that has already learned connections between visual features and scene tags such as "cats," "cars," and "people." The faces and scenes extracted from the photos are displayed on an AGI (ACT-R Graphical Interface) screen as "texts" to make the ACT-R model observe the photo.

Declarative Module

Figure 1a presents examples of declarative chunks. In the examples, *** represents arbitral strings for labeling each chunk, and $< GUID >$ corresponds to photo ID. The top three chunks represent the meaning of the texts displayed on AGI. The bottom three chunks represent attributes of photos. We coded four types of attributes corresponding to "What," "Who," "Where," and "When," following a psychological study of autobiographical memory (Wagenaar, 1986).

The "Who" attribute Using iPhoto face recognition, the two types of chunks signified in *I* and *IV* in Figure 1a were constructed. The chunk *I* associates the text "face753" with a face whose ID is 753. The chunk *IV* states that the photo with ID $< GUID >$ includes face753. The first type of chunk is used to recognize a face from a text on the AGI display. The second type is used to retrieve a photo that includes a recognized face.

The "What" attribute The two types of chunks were constructed from outputs of ReKognition API. As in the case of faces, the chunk *II* in the Figure 1b is used to recognize a scene from the AGI screen. The chunk *V* is used to retrieve a photo including s-broom.

The "Where" attribute This attribute signifies the geographical locations in which the photo was taken. Although recent digital photos have geotag information embedded in their Exif metadata, symbolization of continuous values of latitude and longitude is needed to construct the where attribute for ACT-R. We used the x-means clustering algorithm

to symbolize the location data. Once clusters are made, each location is encoded into the chunk *VI*. The geo30 in the chunk represents a cluster ID. Using this chunk, the model retrieve a photo that shares a geo cluster with the current photo.

The "When" attribute Like the where attribute, the when attribute can be constructed by clustering date-time information embedded in the Exif metadata (the chunk *VII*). In the case of the when attribute, we used k-means with the number of clusters determined by x-means for the where attribute to uniform resolutions of the two attributes.

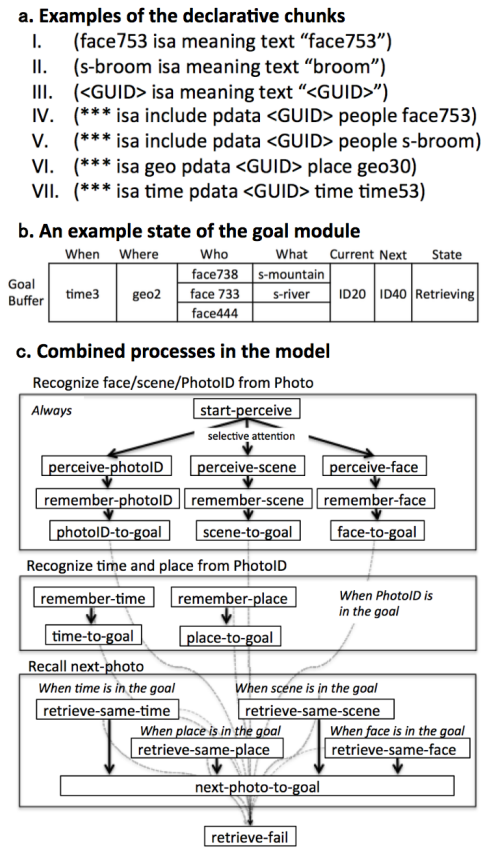


Figure 1: The ACT-R model of autobiographical memory constructed from a photo Library

Goal Module

The ACT-R's goal module is used to hold information associated with the four attributes recognized from the photos. It also holds the current photo ID (the *current-photo* slot), next photo ID (the *next-photo* slot), and the state of the model (the *state* slot). Figure 1b shows an example state of the goal module. The slots for the who and what attributes are defined as pushdown stacks with a size of three.

Production Module

Using the visual input and the declarative memories explained above, the model observes information on the current photo, and retrieves the next photo from a declarative memory. This process corresponds to a free recall of episode memory.

Figure 1c shows the production rules of the model as 19 sets of boxed texts. The rules make up several independent *processes* where arrows connect them. When any process terminates, the state slot of the goal module is changed to *start-process*, and the next process begins. The process has different triggering conditions described as *"always"* or *"when..."* on the figure. When the several processes simultaneously have conditions that correspond to the current state, the current model randomly starts one of the processes. In the future implementation, feedbacks from users would enable the modulation of the probability of a process being selected.

The box at the top of the figure includes processes recognizing visual objects on the AGI screen. The start-perceive rule starts this process by randomly choosing one set of texts from the screen. The ensuing three rules, *remember-photo-ID*, *remember-scene*, and *remember-face*, retrieve chunks that connect visual texts to corresponding chunks (the chunks *I* to *III*). If the retrieval succeeds, the model places the chunk on the corresponding slots of the goal module (*photoID-to-goal, scene-to-goal, face-to-goal*). The middle box in the figure corresponds to the recognition process for the where and when attributes. This information does not directly on the screen. We assume that these attributes are recognized from an intrinsic visual feature of the photo, namely the photo ID. When the current photo ID is in the goal buffer, the *remember-time* and *remember-place* rules retrieve chunks that describe date-time (the chunk *VII*) and location (the chunk VI), respectively. The box at the bottom indicates the retrieval process for the next photo. Each process is triggered by the state of the corresponding slots of the goal module. The tag information in the goal module is used as a query to retrieve chunks including photo ID (the chunks *IV* to *VII*). Once these chunks are retrieved, the retrieved photo ID is stored in the next-photo slot of the goal module.

The model repeats these processes in a *trial*, with the same photo is presented on the screen. During the trial, every time the processes in the boxes at the top and in the middle are triggered, the slots in the goal module are filled. The next-photo slot of the goal module also changes every time the processes in the bottom box are triggered, but it decides which photo to present in the next trial at the end of each trial. Thus, the longer the duration of the trial or the faster the cycle of the model, the richer the information for the retrieval of photos becomes.

Summary

This document described how an ACT-R model of autobiographical memory could be developed from a photo library. In the future paper, we will describe the subsymbolic computation of the model. We are currently developing an interface modulating model parameters by user's behavioral and physiological reactions.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent Tutoring Systems. *Science*, *228*(4698), 456–462.

Chartrand, T. L., & Bargh, J. A. (1999). The chameleon effect: The perception-behavior link and social interaction. *Journal of Personality and Social Psychology*, *76*(6), 893–910.

Routledge, C., Wildschut, T., Sedikides, C., & Juhl, J. (2013). Nostalgia as a Resource for Psychological Health and Well-Being. *Social and Personality Psychology Compass*, *7*(11), 808–818.

Schacter, D. L., Addis, D. R., & Buckner, R. L. (2007). Remembering the past to imagine the future: the prospective brain. *Nature Review Neuroscience*, *8*(9), 657–661.

Tulving, E. (1985). Memory and consciousness. *Canadian Psychology/Psychologie canadienne*, *26*(1), 1–12.

Wagenaar, W. (1986). My memory: A study of autobiographical memory over six years. *Cognitive Psychology*, *18*, 225–252.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Modeling of Proximity-Based Expectations

Stefan Lindner (stefan.lindner@campus.tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems TU Berlin
Berlin, Germany

Nele Russwinkel (nele.russwinkel@tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems TU Berlin
Berlin, Germany

Abstract

Expectations play a crucial role in many domains, including HMI. In this paper we examine a specific type of expectations resulting from the proximity of interface control elements. We briefly present the results of an experimental smart phone task that manipulated the relationship between control element proximity and the closeness of the corresponding goals. We present a modeling approach for proximity-based expectations and compare model predictions from an ACT-R model and experimental results.

Keywords: expectations, interface design, cognitive modeling, ACT-R, HMI

Introduction

Expectations are hugely important in everyday life. They are an important element of learning about, dealing with and ultimately mastering our environment. More specifically expectations allow us to anticipate future states of the environment. This allows both for better mental and action preparation (Umbach et. al. 2012) but also for improved action-feedback learning loops (Friston & Kiebel 2009, Gallistel, 2005).

In the case of proximity and causality a type of expectation might have evolved that lead us (largely subconsciously) to expect similar or close objects in our environment to be functionally or causally related. We will refer to them as *proximity-based expectations*. Modeling these expectations is an important puzzle piece in the quest towards making quantitative predictions about usability. By quantifying their exact impact, we can improve future models of user interaction with technical interfaces by adding expectations to them.

We created a cognitive modeling approach in ACT-R that utilizes one of the possible implementations of proximity-based expectations and compared its output with experimental data.

We also devised an experimental setup that aims to empirically capture the effect of a specific design decision - here spatial proximity of control elements - on reaction times and user errors. To this purpose we created a smart phone app that enabled the construction and configuration of geometrical shapes.

Experiment

Participants were asked to recreate three geometrical shapes of varying shape, filling color and periphery color. Each trial started with the app presenting a screen that contained the three shapes to be recreated and buttons that could be used to initiate the manipulation of each shape (see figure 1, left panel).

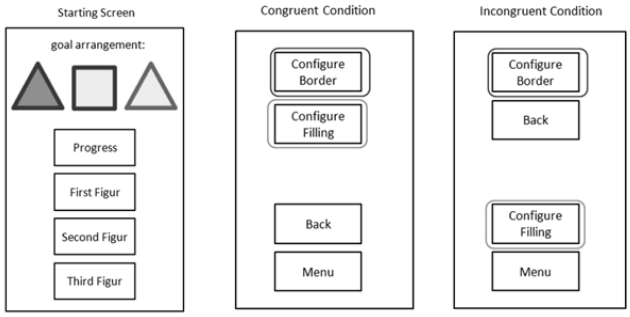


Figure 1: Starting screen (left) and exemplary menu state for the congruent condition (left) and the incongruent condition (right).

The participants first had to choose the shape (square or triangle) and then had the choice between manipulating the color of periphery or interior. The participants were given constant feedback about the state of the shape so that they could track the effects of their manipulations. The menu always contains four buttons, with two spatially close buttons on the top and the bottom of the screen respectively.

We tested two experimental conditions: in the "congruent condition" - in accordance with the proximity compatibility principle - buttons that were used for similar purposes (like the button for manipulating shape and the button for manipulating color in figure 1, middle panel) were situated close to each other. Conversely, in the "incongruent condition", buttons for similar purposes were situated away from each other (figure 1, right panel). The participants had to finish a total of ten trials, each starting with the presentation of the three shapes and ending with the correct creation and configuration of all three shapes.

ACT-R expectation model

Two model approaches for modeling proximity-based expectations were proposed by Lindner & Russwinkel (2015). In the current paper we will present one of them, the action tendency approach).

Both for the model implementation and the concept description in this paper we made use of the cognitive architecture ACT-R (Anderson et al., 2004) and its terms, respectively.

The goal of the modeling approach is to quantify both the processes involved in building up expectations and those that translate those expectations into changes in overt behavior. The main idea consists in linking co-occurring goals and actions and to create action tendencies from these links.

In the experimental task let us assume that a participant has the goal to configure the border of a shape and then successfully does so by pressing the button „configure border". The button itself but also the buttons close to the button „configure border" should from now on be associated with the goal „configure the border". They should also be associated with related goals like "configure the shape" (which is a meta-goal of „configure the border") and "configure filling" (which is a sub-goal of this meta-goal).

More technically speaking, if a goal/sub-goal G is achieved by using control element E the following processes occur: First, the elements close to E, including E itself, C(E) are associated with the goals close to G, including G itself, C(G) (e.g. sub-goals, sub-goals of the meta-goal) (see figure 2). Second, action tendencies are created that "encourage" the use of elements from C(E) when the goals from C(G) or reoccur.

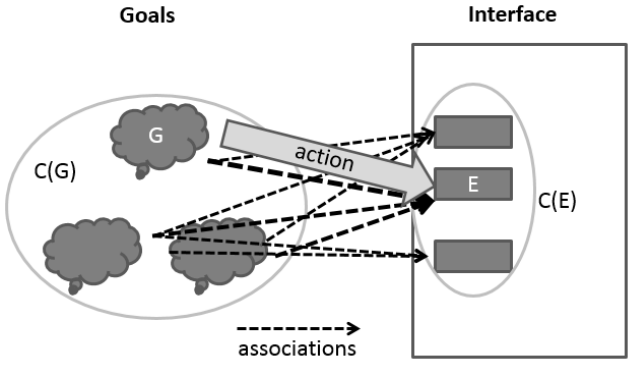


Figure 2: Associations between closely related goals and spatially close control elements

The "action tendency" implementation comprises the direct and immediate creation of all specific action tendencies related to the current goals and interface elements. In ACT-R this translates into the creation of precise productions that couple the present goal and related goals with spatially close control elements anytime a control element is successfully used. The starting utilities of the productions (and thus the probability of them being used) grow with closeness to the original goal G and spatial closeness to the original control element E.

So far, in ACT-R production utilities only change after a reward is given at the end of a successful action sequence that contained the production. In order to implement the utility needed for our expectation approach, we had to extend the utility mechanism to also include the change of utilities of production that were not previously fired.

One important implementation decision is what the action tendency actually entails. In our model we conservatively stuck the interpretation that the participant will first look to an expected position when encountering a new screen. They will also prepare to press the "expected" button before it is visually encoded.

General Model Predictions

The expectations will lead to more frequent visual encoding of the correct control element first if the interface is constructed following PCP. This should result in overall faster completion time of tasks. On the one hand, visual search is cut short if the expectation already points to the correct control element. On the other hand, the motor preparation for the expected button should also lead to a decrease in motor action, as both movement and motor preparation can be skipped.

We also expect fewer errors to be committed in an interface following PCP compared to one that does not. Our model, however will not address this hypothesis. In the discussion we will elaborate on a model extension that could reflect this phenomenon.

Experimental and modeling results

	n	Model w/o expectations	Model w/ expectations	Experiment
Congruent Condition	19	29,2	29,0	29,3
Incongruent Condition	17	29,5	29,6	36,9

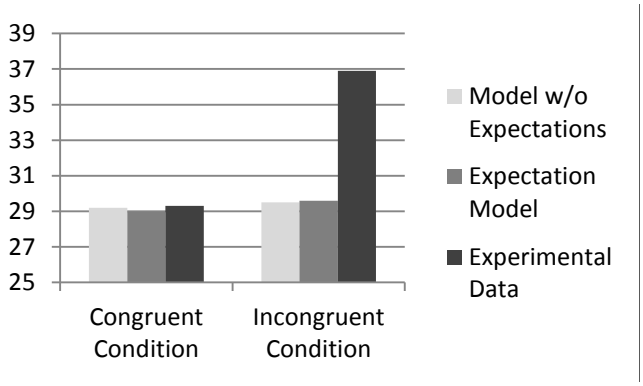


Figure 3: Total Completion Times in s (first trial excluded)

Discussion and Outlook

In order to better reflect experimental reaction times - especially in the incongruent condition - the model is currently being altered to include the tendency to click screen elements that are expected to be helpful for the task without double checking its function first. This could also help to better fit the experimental results concerning errors committed, since participants committed a substantial overall amount of errors and errors were more frequent in the incongruent condition.

References

Anderson, J.R., Bothell, D., Byrne, M.D., Douglas, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111, 1036-106.

Friston,K. & Kiebel, S. (2009). Predictive coding under the free-energy principle. *Philisophical Transactions of the Royal Society, Biological Sciences. 364*, 1211-1221.

Gallistel, C. R. (2005). Deconstructing the law of effect. *Games Econ. Behav. 52*, 410–423.

Lindner, S. and Russwinkel, N. (2015). The Effect of Design Decisions on User Expectations - A modelling approach. *Berliner Werkstatt Mensch-Maschine-Systeme, 2015*.

Umbach, V. J., Schwager, S., Frensch, P. A., & Gaschler, R. (2012). Does explicit expectation really affect preparation?. *Frontiers in psychology, 3*.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

A Proposed Method of Matching ACT-R and EEG-Data

Sabine Prezenski (sabine.prezenski@tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems, Technische Universität Berlin
10587 Berlin, Germany

Nele Russwinkel (nele.russwinkel@tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems, Technische Universität Berlin
10587 Berlin, Germany

Keywords: ACT-R; EEG; ICA

Introduction

Most model-based research in neuroscience is limited to fine grained analysis of single cognitive process. The question how different brain regions interact with each other is a matter of ongoing research and far from being answered. Methods, which unite findings and methods of cognitive modeling and neuroscience are required, in order to obtain greater understanding of cognitive processes of the human brain. The objective of this paper is to propose a matching method that links Independent Components (ICs) derived from EEG-data (Electroencephalography) and ACT-R buffer activation, using dipole fitting and cross-correlation analysis.

Theory

ACT-R

ACT-R is a cognitive architecture, which consists of different modules (visual, goal, declarative, imaginal, motor, procedural and others). Buffers are the interfaces of the modules and interact via a (rule-based) production system. ACT-R has the advantage that human cognition, as a whole, in contrast to other approaches that only focus on single cognitive processing steps, is modeled. Nevertheless, predictions of perception, action and cognition steps are provided by ACT-R models, in the range of milliseconds.

FMRI & ACT-R

There is growing research that combines ACT-R theory with findings from neuroimaging techniques. In recent years many studies have emerged that compare ACT-R module activity with brain activity as measured in fMRI (functional magnetic resonance imaging) studies. The combination of ACT-R and fMRI provides information about a) the localization of specific modules (e.g. procedural components in basal ganglia) and b) the plausibility of the architecture (e.g. reasonability to assume a specific module) (Anderson, 2008). But, information concerning the exact timing of cognitive processes cannot be obtained with fMRI.

EEG and ACT-R

EEG (electroencephalography) measures voltage fluctuations of neurons. Thus, it reflects a large amount of ongoing brain processes. Different components of EEG activity are linked to different information processing stages and different parts of the brain. EEG has a very high temporal (milliseconds) resolution. The low spatial resolution of EEG data is a disadvantage compared to FMRI data. But methods like Independent Component Analysis (ICA) possibly hold the potential to overcome this limitation (Delorme, Palmer, Onton, Oostenveld & Makeig, 2012). ICA separates a set of mixed signals into their respective source. ICA determines which temporally independent and spatially fixed activations make up a time-varying response. By using ICA, it is possible to approximate where signal components originate and separate the mixed signals on into different components Thus, theoretically the source of activation can be found.

Only few studies attempted to combine ACT-R models with EEG data; even though EEG and ACT-R allow milliseconds precise information about the timing of processes, whereas fMRI does not.

Two EEG-ACT-R studies have merely dealt with timing issues of certain request (Cassenti, Kerick & McDowell, 2011; Cassenti, 2007). Two different studies have applied ICs in order to gain information about module activation, but both had severe methodological limitations since they measured EEG with only five channels (Griffiths & West, D'Angiulli, A., 2011) or only one subject (Prins, 2010). In a more promising study, Van Vugt investigated coherences between EEG- frequency bands and buffer activity (Van Vugt, 2014). She found a link between theta frequency bands and working memory modules of ACT-R. Recently, semi-hidden Markov-models were used on EEG data to indentify the number and duration of cognitive processing stages (Borst & Anderson, 2015; Anderson, Zhang, Borst, & Walsh, 2016). These processing stages were then qualitative compared to the predictions of different theoretical models, including an ACT-R model.

Objective

Our goal is to develop a method, which directly combines ACT-R activation and ICs of EEG-activation. The main advantage is using the potential of EEG-data for qualitative and quantitative model validation. In order to assess whether the ACT-R formalized theory represents real human behavior, experimental data is collected. Therefore, participants perform the same task as the model. To validate the modeled data, the matching of modeled data and participant data is assessed via Goodness of Fit Indices. In general, behavioral data (reaction time, mistakes) is consulted for this assessment. If the overall fit of a model

reproduces the data well, this is seen as an indication that human information processing, as formalized in the model, could well proceed in this way. However, it is possible, that models postulating different processing steps, achieve a similar match to behavioral data. Using EEG-data could allow process validation of models. If the ICs were to match to specific model components then the timing of peaks of buffer activity should match IC-peaks. In order to discover which aspects of human information processing are linked to which EEG components is another important advantage of the proposed method.

Matching Method

The following section outlines the main steps:

The *first step* is to find an experimental paradigm, meeting the following demands: Ideally, it is a well studied paradigm, for which high-quality EEG-data, with a decent spatial resolution and ACT-R models exist. A further requirement for the paradigm is that, it produces activity over the neocortex, preferably in well separated areas and that the ACT-R model utilizes different ACT-R modules.

The *second step* is then to fit the model-data to the behavioral-data of the EEG-study. The duration for the task for the model should match the average duration of the participants (e.g. r^2 should be above 0.800 and RSME should be small). Parameters of the model could be adjusted in order to achieve a fit satisfying these constraints.

The *third* step is then to use a linear transformation to scale the model activity on a trial-by-trial so that the behavioral data and the model data achieve a perfect fit. Such a procedure was introduced by Borst, Taatgen and Van Rijin, 2011 in order to match fMRI to ACT-R data.

The *forth step* requires displaying ACT-R buffer activity in milliseconds. This can be achieved by averaging data from multiple model runs for each buffer, as done by Van Vugt (2013). In summary, data from multiple model runs is averaged, using as many model runs as trials in the EEG data. In order for the modeled and the EEG data to have the same amount of data points, the average module activation of each buffer needs to be sampled to the EEG sampling rate. The *fifth step* concerns the EEG-data. The EEG-data needs to be transformed into Independent Components (ICs). This will result in as many components as there are EEG channels and thus more ICs then modules. For a cross-validation (see step sixth) dipole fitting (see Griffiths and West, 2011 for an example of how this method works in combination of ACT-R) should be deployed. With dipole fitting the sources of the ICs in the neocortex can be found.

The *sixth step* involves the actual matching of EEG and ACT-R data. A correlation method should be applied. Cross-correlation analysis is a promising approach, as it can be used to find correlations between matrices.

After ICs that match buffer activation are found, *the seventh step* proceeds to cross-validate the components using dipole fitting; the sources of the selected ICs should be located in brain regions that have been identified to locate the ACT-R modules (Anderson, 2008).

The *eigth step* analyses and describes ICs that are associated with different buffers.

Finally, as a *ninth step*, these previously identified characteristics will be used on different data.

Discussion & Outlook

Taken together, research in the field of linking ACT-R and EEG-data is so far limited. But, important findings, concerning cognitive theory and its neural correlates, could derive from combining data of ACT-R models and EEG-activation.

To implement the proposed matching method, we are searching for a suitable paradigm. The main requirement is that an ACT-R model for this paradigm uses numerous ACT-R modules, with uncorrelated activation and that EEG data exist.

References

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* New York: Oxford University Press.

Anderson, J. R., Fincham, J. M., Qin, Y., and Stocco, A. (2008). *A central circuit of the mind*. Trends in Cognitive Sciences, 12(4), 136-143.

Anderson, J. R., Zhang, Q., Borst, J. P., & Walsh, M. M. (2016). *The Discovery of Processing Stages: Extension of Sternberg's Method.* Psychological Review.

Borst, J.P., Anderson, J.R (2015). *The discovery of processing stages: Analyzing EEG data with hidden semi-Markov models.* NeuroImage 108, 60-73.Borst, J. P., Taatgen, N. A., & Van Rijn, H. (2011). *Using a symbolic process model as input for model-based fMRI analysis: Locating the neural correlates of problem state replacements.* Neuroimage, 58(1), 137-47.

Cassenti, D. N. (2007). ACT-R Model of EEG Latency Data, In *Proceedings of the Human Factors and Ergonomics Society 51st Annual Meeting* (pp. 812-816). Santa Monica, CA: Human Factors and Ergonomics Society.

Cassenti, D. N., Kerick, S. E., & McDowell, K. (2011). *Observing and modeling cognitive events through event-related potentials and ACT-R.* Cognitive Systems Research, 12(1), 56-65.

Prins, H. (2010). *Comparison between EEG data and ACT-R buffer activity during the Attentional Blink using Independent Component Analysis*. Bachelor Thesis: University of Groningen

Griffiths, G. D., West, R., & D'Angiulli, A. (2011). Cognitive modeling of event-related potentials. In *Proceedings of the 33th Cognitive Science Society Annual Meeting* (pp. 1794-1798). Boston, USA.

Delorme A., Palmer J., Onton J., Oostenveld R., Makeig S. (2012). *Independent EEG Sources Are Dipolar*. PLoS ONE 7(2): e30135.

Van Vugt, M. K. (2013). Towards a dynamical view of ACT-R 's electrophysiological correlates. In: P*roceedings of the 12th International Conference on Cognitive*

Modelling (pp. 11-16). Ottawa, Canada: Carleton University.

Van Vugt, M. K. (2014). *Cognitive architectures as a tool for investigating the role of oscillatory power and coherence in cognition.* NeuroImage, 85, Part 2(0), 685–693.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Interactions of Declarative and Procedural Memory in Real-Life Tasks: Validating CPR as a New Paradigm

Florian Sense (f.sense@rug.nl)
Department of Experimental Psychology and Department of Psychometrics and Statistics, Groningen, The Netherlands

Sarah Maass (s.c.maass@rug.nl)
Research School of Behavioral and Cognitive Neuroscience, Groningen, The Netherlands

Hedderik van Rijn (d.h.van.rijn@rug.nl)
Department of Experimental Psychology and Department of Psychometrics and Statistics, Groningen, The Netherlands

Keywords: declarative memory; procedural memory; complex skill; new paradigm

Introduction

In the learning and memory literature, there is a clear distinction between procedural and declarative memory. We know that they develop differently in children (Finn et al., 2016) and that they are dissociated anatomically and this distinction is reflected in cognitive architectures as well (Anderson et al., 2004). In the lab, most tasks tap into and measure either one of those components. In most complex real-life skills, however, both declarative and procedural memory are required to perform well.

Here we will look at the learning of a complex real-life skill: cardiopulmonary resuscitation (CPR). More specifically, we will teach participants *basic life support* skills meant to be performed on an adult victim suffering from cardiac arrest. CPR has both declarative and procedural components and there are clear guidelines prescribing the sequence of steps that need to be executed (Perkins et al., 2015). This sequence needs to be remembered (e.g., *I need to call am ambulance* before *I initiate CPR*) which draws on declarative memory. Learning to administer correct compressions, on the other hand, is probably closer to a procedural skill.

CPR is an ideal task for various reasons. It is clearly constrained and can be performed and monitored in a controlled (lab) environment while staying close to how it would be trained in a real-life setting. Publicly accessible guidelines provide clear learning criteria against which the obtained measures can be compared. The skill can be learned in a single session and pilot data suggest that there are individual differences in how quickly CPR performance decreases after initial learning and that not all aspects of CPR are retained equally well. Therefore, CPR is an ideal testbed to investigate the interaction between declarative and procedural learning in a real-life setting. In this exploratory study, we will investigate which aspects of CPR performance are best predicted by someone's declarative learning or procedural learning ability.

Tasks

To address the research question, we will have each participant perform three tasks. Each task is intended to measure one of the three components of interest: acquisition and performance of CPR, procedural learning, and declarative learning.

Learning CPR

Each participant is taught "adult basic life support" skills in accordance with the guidelines of the European Resuscitation Council (Perkins et al., 2015). Hereto, participants watch an instructional video and then practice CPR with a Laerdal Resusci Anne QCPR manikin. Data is recorded using the Laerdal SimPad SkillReporter. This setup allows detailed recordings of the skill development during the acquisition of basic life support skills: both the order of the steps that were taken can be recorded as well as detailed measures of each compression and rescue breath that is administered. After an initial training phase (with corrective feedback), CPR performance will be assessed about 45 minutes later and either one or four weeks later.

Procedural Learning

To assess a participant's expertise in procedural learning, we selected separate implicit and explicit procedural knowledge tasks.

Serial Reaction Time Task. In the serial reaction time task (SRTT) visual cues appear over four response options and the participant needs to respond with one of four fingers that are mapped to the response options. The task can be used as a measure of implicit procedural learning by presenting the participant with different blocks of trials: one block in which the order of responses is random and one in which the order is a repeating sequence (Robertson, 2007). The participant is (usually) not aware of the sequence but their reaction times become markedly faster. A learning measure can be computed by subtracting the mean reaction time in the sequenced block from the mean in the random block (e.g., Willingham, Salidis, & Gabrieli, 2002).

Mirror Tracing. In the mirror tracing task, the participant sees a line with 12 corners connecting two points and needs to trace the line. However, they can only see the line and their own hand through a mirror. With practice, both the completion time and the number of errors decreases, indicating procedural learning. Quantifying the improvement provides a learning measure that is linked to individual differences (e.g., Finn et al., 2016).

Declarative Learning

To derive a learning measure for someone's ability to learn declarative information, we use a fact learning system developed in our lab. The system uses retrieval practice to quiz learners on a trial-by-trial basis. This allows recordings of accuracy and response latencies which are used to estimate the current memory strength of the test item, which are used to schedule a repetition of the item before it is forgotten (Van Rijn, van Maanen, & van Woudenberg, 2009). As more information is gathered, an estimate of how quickly an item is forgotten is fine-tuned. After studying 35 Swahili-English word-pairs for 15 minutes, an estimated rate of forgetting can be obtained for each participant. This learning measure indicates how quickly, on average, a participant forgets this type of material. We have shown that this measure can be measured reliably (Sense, Behrens, Meijer, & Van Rijn, 2016) and is not related to common measures of executive attentional functioning (Sense, Meijer, & Van Rijn, accepted).

Procedure

A total of 40 participants with no prior CPR experience will be invited to participate. So far, data from 20 participants has been collected. Participants will be invited for two sessions. In the first, they will be taught CPR and will complete the 15-minute word-learning session, 768 trials of the SRTT (half random, half sequenced), and nine mirror-tracing trials. At the end of session one, they will be tested on the CPR performance. Either one or four weeks later, in session two, participants will be tested on the Swahili words they learned as well as on their CPR performance. Also, they will complete another round of 768 SRTT and nine mirror-tracing trials.

Planned Analyses

This is an exploratory study to verify whether the acquisition and forgetting of CPR skills can be captured in this paradigm and how CPR performance is related to established declarative and procedural learning tasks.

The computation of learning measures for the declarative and procedural tasks is straightforward, however, valid measures for CPR still need to be developed. As a large number of measures are recorded, the main challenge will be to construct suitable learning measures. We will report a first exploration of various options and discuss their merits.

The derived learning measures can then be correlated with the learning measures for the declarative and procedural tasks. Multiple regression can be used to determine which (combination of) measures can best explain variance in overall CPR performance. Furthermore, we will look at which aspects of CPR are forgotten more quickly than others and lead to a decrease in overall CPR performance.

Hopefully, this project will provide useful information with regards to CPR's relationship to declarative and procedural learning as well as indications that might be helpful to optimize (re-)learning of CPR. Furthermore, steps will be taken to establish CPR as a complex real-life task that involves both declarative and procedural components. As such, modeling human behavior in this task can shed more light on the relative contributions of declarative knowledge and procedural skills in complex human behavior.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036–60.

Finn, A. S., Kalra, P. B., Goetz, C., Leonard, J. A., Sheridan, M. A., & Gabrieli, J. D. E. (2016). Developmental dissociation between the maturation of procedural memory and declarative memory. *Journal of Experimental Child Psychology*, *142*, 212–220.

Perkins, G. D., Handley, A. J., Koster, R. W., Castrén, M., Smyth, M. A., Olasveengen, T., … Greif, R. (2015). European Resuscitation Council Guidelines for Resuscitation 2015. Section 2. Adult basic life support and automated external defibrillation. *Resuscitation*, *95*, 81–99.

Robertson, E. M. (2007). The Serial Reaction Time Task: Implicit Motor Skill Learning? *The Journal of Neuroscience*, *27*(38), 10073–10075.

Sense, F., Behrens, F., Meijer, R. R., & Van Rijn, H. (2016). An Individual's Rate of Forgetting is Stable over Time, but Differs Across Materials. *Topics in Cognitive Science*, *8*(1), 305–321.

Sense, F., Meijer, R. R., & Van Rijn, H. (accepted). On the Link between Fact Learning and General Cognitive Ability. In *Proceedings of the 38th Annual Meeting of the Cognitive Science Society*.

Van Rijn, H., van Maanen, L., & van Woudenberg, M. (2009). Passing the test: Improving learning gains by balancing spacing and testing effects. In *Proceedings of the 9th International Conference on Cognitive Modeling* (pp. 110–115).

Willingham, D. B., Salidis, J., & Gabrieli, J. D. E. (2002). Direct comparison of neural systems mediating conscious and unconscious skill learning. *Journal of Neurophysiology*, *88*(3), 1451–1460.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Intuitive Decision-Making Revisited: A Heuristic and the Feeling of Recognition

William G. Kennedy (WKennedy@GMU.Edu)
Krasnow Institute for Advanced Study, George Mason University, 4400 University Drive
Fairfax, VA 22030 USA

Abstract

At a previous International Conference on Cognitive Modeling (ICCM) a simple model of intuitive decision-making was presented. The task was to learn and then recognize strings that had a hidden structure. The model did well a matching of human performance on hits, misses, correct rejections, and false alarms. A deeper look reveals not just more details about the context and challenge of the memory task, but an explanation of the associated heuristic and the feeling of recognition.

Keywords: Intuitive decision-making; Recognition Heuristic; feeling of recognition.

Introduction

With and without conscious effort we train our subconscious mind and we can use that learning to improve performance on explicit tasks (Lehrer, 2010). This is the fundamental idea behind the bestselling books on the topic each describing many examples of the phenomena (Gladwell, 2007; Gigerenzer, 2007). Intuitive decision-making refers to implicit pattern recognition that is not thought to involve symbolic rules (Klein, 1998).

The ACT-R theory (Anderson, 2007; Anderson, et al., 2004) represents memory tasks by the building activations for the discrete, symbolic things we want to remember. The theory and architecture compares the activation of items in memory against a threshold to determine whether a retrieval attempt is successful thus making of remembered item consciously available.

A previous ICCM conference paper (Kennedy & Patterson, 2012) described a model of the process on an intuitive learning task (Reber, 1967), i.e., below the level of individual object recognition. The idea was that instead of training increasing the activation for the discrete items to be learned, another process was taking place, which noted the structure of the objects to be learned. This deeper, unconscious, intuitive learning supported the performance at the higher, explicit level. The model did very well at matching the human subjects' performance, the hits, misses, correct rejections, and false alarms. See Table 1 for updated results.

Table 1: Human and Model Performance.

Response type	Human (SEM)	Model
Hits	31.5/44(2.7)	34/44
Misses	12.5/44(2.2)	11/44
Correct Rejections	35.6/44(2.7)	39/44
False Alarms	8.4/44(2.2)	5/44

The learning and retrieval process described in the previous paper appears to be related to the Recognition Heuristic described by Gigerenzer's group (Gigerenzer, Todd, & the ABC Research Group, 1999; Gigerenzer, Hertwig, & Pachur, 2011). That Recognition Heuristic relies on a discriminatory level of recognition: one of two choices being recognized, the other not. The selection criterion is useful because it is often correlated with recognition goal. The algorithm implemented in the ACT-R model was to consider each sequential pair of letters, a bigram, in turn through to the end and for each bigram to decide. If the bigram is recognized, the next one is considered. If not, the string is not recognized. If the process reaches the end and each bigram had been recognized, then the string is considered recognized and therefore presumed to be valid.

Here I present a deeper description of the task, the training, the testing, and the performance of the human subjects and the model providing additional support for the capabilities of the ACT-R architecture to represent intuitive learning and performance with some effort by the modeler.

Deeper into the Task

At one level, the Reber task is a standard memory task with training (presentation of the objects to be remembered) followed by tests of recall of those and similar objects. However, the purpose of the scenario is not the explicit memory for the specific objects used in the training, but the development and testing of the patterns within the objects. The objects, specifically, strings of letters, have the structure presented in Figure 1. The subjects are not shown nor explicitly trained on the structure itself, but it is implicitly presented through the strings presented in training. The testing evaluates the learning of the underlying structure because the training does not present the full set of the strings to be recognized.

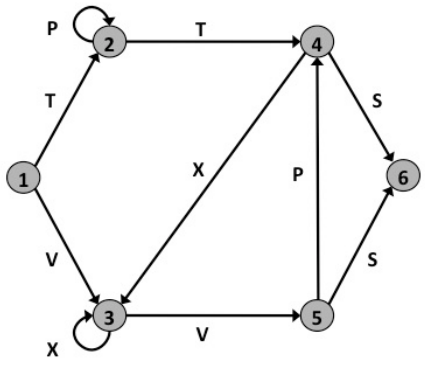


Figure 1. Finite state diagram defining a grammar of letter strings. (From Reber, 1967.)

Deeper into the Training

The training protocol presents a randomly selected set of 18 strings of length 8 or fewer. However, what needs to be learned is the structure. The ACT-R model made the structure explicit by noting the sequencing of the letters in the training strings as bigrams. In the model, each training string is treated as a series of bigrams to be learned as discrete objects (chunks) for which activations were developed.

Analysis of the grammar determined that the full set of 40 valid strings is made up of only 14 bigrams. As an example, the strings "TPPTS", "TPPPTS", "TPPPPTS", and "TPPPPPTS" are made up of only 4 bigrams: "TP", "PP", "PT", and "TS". Analysis of the number of repetitions of the 14 bigrams in 1,000 training sets of 18 strings found that all the bigrams are likely to be presented to each participant (although some with high variance) even though less than half (41%) of the full strings are included in the training.

Deeper into the Testing

The testing protocol used 22 randomly selected valid strings and 22 randomly generated invalid strings using the same letters. Each member of the test set is presented twice resulting in 44 possible hits/misses and 44 possible correct rejections/false alarms. Analysis of 1,000 sets of 22 valid strings revealed that all of the bigrams were used, but again the more rare ones having high variance.

Deeper into the Performance

The performance of the human subjects and the model can be discussed below the string level as well. The data available on the human subjects includes the specific training sets and testing sets, with performance on the test set at the individual string level. The analysis of the human and model's performance at the string and bigram levels shows the strings for each type of response (hits, misses, correct rejections, and false alarms) are very similar.

Discussion

The similarity of the performance on this task at the deeper level is further evidence that the model and the human subjects are using the same process (heuristic) to intuitively learn and decide the questions in this task. This is considered relying on intuitive or "gut" feelings because the only thing used in the ACT-R memory retrieval process is only the status of whether the retrieval was successful or not. This is a new, beyond rational representation of cognition already supported within the ACT-R theory and architecture, although the topic is not new (Lebiere & Wallach 2001). It also supports the concept of the transfer of basic cognitive skills below the symbolic level of ACT-R (Taatgen, 2013). As such, it has the potential to represent many of the variety of intuitive decisions we make every day.

Acknowledgments

This work was originally funded in part by AFOSR/AFRL grant FA9550-10-1-0385 and the George Mason University Center of Excellence in Neuroergonomics, Technology, and Cognition (CENTEC). The deeper investigation was partially funded by the Center for Social Complexity within the Krasnow Institute for Advanced Study at George Mason University.

References

Anderson, J. R. (2007). How Can the Human Mind Occur in the Physical Universe? Oxford: Oxford University Press.

Anderson, J. R., Bothell, D., Byrne, M.D., Douglas, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of mind. *Psychological Review, 111*, 1036-1060.

Gigerenzer, G. (2007). *Gut feelings: The intelligence of the unconscious*. Penguin.

Gigerenzer, G., Hertwig, R., & Pachur, T. (2011). *Heuristics: The foundations of adaptive behavior*. Oxford University Press, Inc.

Gigerenzer, G., Todd, P. M., & the ABC Research Group (1999). *Simple heuristics that make us smart*. Oxford University Press, USA.

Gladwell, M. (2007). *Blink: The power of thinking without thinking*. Back Bay Books.

Kennedy, W. G., & Patterson, R. E. (2012). Modeling intuitive decision making in ACT-R. *Proceedings of the 11th International Conference on Cognitive Modeling* (pp. 1-6). Berlin, Germany.

Klein, G. (1998). Sources of power: How people make decisions. Cambridge, MA: MIT Press.

Lebiere, C. & Wallach, D. (2001). Sequence Learning in the ACT-R Cognitive Architecture: Empirical Analysis of a Hybrid Model. In R. Sun & C. L. Gilles (Eds.). Sequence Learning: Paradigms, Algorithms, and Applications (pp. 188-212). Berlin: Springer Lecture Notes in Computer Science.

Lehrer, J. (2010). *How we decide*. Houghton Mifflin Harcourt.

Reber, A. S. (1967). Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior, 6*, 855-863.

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological Review*, *120*(3), 439.

Wallach, D. & Lebiere, C. (2003). Implicit and explicit learning in a unified architecture of cognition. In L. Jimenez (Ed.) Advances in Consciousness Research, 215-250. Amsterdam: John Benjamins Publishing Company.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

A Computational Model of Semantic Convergence in Bilinguals

Shin-Yi Fang (szf4@psu.edu)
Department of Psychology, Pennsylvania State University
University Park, PA 16802 USA
Benjamin D. Zinszer (bzinser@gmail.com)
Department of Brain and Cognitive Sciences, University of Rochester
Rochester, NY 14627 USA
Barbara C. Malt (barbara.malt@lehigh.edu)
Department of Psychology, Lehigh University
Bethlehem, PA 18015, USA
Ping Li (pul8@psu.edu)
Department of Psychology, Pennsylvania State University
University Park, PA 16802 USA

Abstract

Patterns of object naming were influenced by the language of the people used. In this study, we aimed to simulate the bilinguals' naming patterns and investigate the underlying mechanisms. Our results replicated the empirical findings that (1) bilingual speakers develop converged naming patterns in their two languages that are distinct from those of monolingual speakers of each language and (2) how the bilingual semantic convergence was manifested in their lexical representations. We also demonstrated that both orthography and connections between two languages are important to establish converged naming patterns in bilinguals. Furthermore, our modeling data suggested that the strength of name agreement in the two languages are involved in the cooperation and competition relationships on the object naming. Our model provides a foundation for examining the factors contributed to the patterns of object naming in bilinguals.

Keywords: object naming; lexical categories; modeling; self-organizing map; bilingual lexicon

Introduction

Across languages, objects are not always been classified into the same categories. For example, Malt, Sloman, Gennari, Shi, and Wang (1999) asked speakers of American English, Argentinean Spanish, and Mandarin Chinese to name 60 common household containers and found that the naming patterns differ substantially as a function of the language spoken. For speakers of two languages, how they dealt with the inconsistent mapping relationships between objects and names could enhance our understanding about language learning. Ameel, Storms, Malt, and Sloman (2005) investigated the naming patterns of adult Dutch-French simultaneous bilinguals, Dutch monolinguals, and French monolinguals. They found that object naming patterns by the bilingual speakers converge toward a pattern that is different from the naming patterns of monolinguals of each language, suggesting that simultaneous bilinguals do not behave like monolinguals in lexical categorization and the bilingual lexical representations are not simply the sum of two separate monolingual representations.

Ameel, Malt, Storms, and Van Assche (2009) further investigated the naming patterns in Ameel et al. (2005) and the typicality ratings between language groups. They examined if the bilinguals' semantic convergence was manifested in the centers and/or at the boundaries of lexical categories. Ameel et al. (2009) found that the translational equivalents have closer category centers in the naming patterns of the bilinguals than those of the monolinguals and the closeness in bilinguals were contributed by more than the boundary exemplars alone. Ameel et al. also found that bilinguals needed fewer dimensions to separate categories than monolinguals, suggesting that the category structures were less complex for bilinguals and that naming were more likely to be based on similarity to the prototypes.

In this study, we build a model based on self-organizing maps (SOM; Kohonen, 2001) to study bilingual object naming. By building and testing this computational model against existing data of Ameel et al. (2005; 2009), this work will provide the foundation to investigate underlying mechanism and further modeling study to manipulate learner characteristics such as age of exposure, frequency of input and proficiency in each language, as well as lexical input variables such as similarities between the lexical items.

Method

We constructed is a multi-layer SOM network, which includes three basic SOMs (i.e., semantic, phonological, or orthographic). The three SOMs are connected via associative links updated by bi-directional Hebbian learning. In addition to the basic SOM architecture, we added lateral between languages in the model to simulate between-language interactions. The lateral connections are implemented with the nodes that are fully connected with each other. The connection weights are updated via Hebbian learning rule. We used the monolingual naming data from Ameel et al. (2005) as the basis of input to the model. We trained the model on representations of pictures of 73 bottle-like objects that are typically named as bottle, jar, or container in American English or else to have one or more

salient properties in common with objects called by those names. We call this the "standard" model.

In order to identify the role of orthographic information and the effects of lateral connections, we constructed two comparison models to contrast with the standard model, respectively. The comparison models are identical to the standard model in every respect, except that in each comparison model one component is removed: Comparison Model A was constructed without orthographic information and Comparison Model B was constructed without lateral connections between languages.

Results

We found that, similar to Ameel et al. (2005), our computational model shows higher correlation between bilinguals' two language (0.97) than between two monolingual languages (0.63), indicating that our model simulated empirical naming patterns and captured bilinguals' lexical categorization. Both comparison Model A and B showed significantly worse performance than standard Model, suggesting that the both orthography and lateral connections were important in the model's object naming. Furthermore, the Model B performed worse than Model A suggesting that the role of lateral connection is more critical than the role of orthography. We also replicated that the main findings in Ameel et al. (2009), indicating that the translational equivalents have closer category centers in the naming patterns of the bilinguals than those of the monolinguals and bilinguals have less complex category structures than monolinguals.

We further examined the model to explore what properties in the model might have influenced the naming patterns. Our model shows that if an object elicited a strong level of activation for a word in the target language, the output name of the model for bilingual naming will be the same as the name for monolingual naming. However, if the activation level is weak in the target language and the cross-activation from the non-target language is strong, the output names of the model could be different between bilingual naming and monolingual naming. For example, if a bottle-like object elicited strong activation of the word *fles* in Dutch, both the monolinguals and bilinguals will produce *fles* in Dutch; whereas the activation of *fles* in Dutch is weak and the activation of *bus* in Dutch outperformed *fles*, due to combination of its original activation from Dutch and the strong lateral activation from French. In this example, the monolinguals will produce *fles*, but the bilinguals will produce *bus*.

Discussion

In this study, we successfully built a bilingual lexical categorization model based on connectionist SOM architecture that has been previously tested in other domains of language acquisition and processing. Our model simulated bilingual semantic convergence in the naming of common household objects as reported in the empirical literature (Ameel et al., 2005; 2009).

Our standard model performed significantly better than the two comparison models in which either lateral connection or orthography components not included. This is particularly important as our model is designed to simulate the dynamic interactions between two languages, and the orthography and the lateral connections play a critical role in bilingual lexical categorization. Our results demonstrate how, for simultaneous bilinguals, the processing of one language can be influenced by the other language (i.e., bi-directional influences between languages). Our simulation also showed that the strength of name agreement is an important factor to determine lexical naming patterns for bilinguals. If the object has high name agreement in one language, the influence from other language through lateral connection cannot easily change its category and vice versa.

The viability of our model paves the way to use modeling to study a wide range of learner and object name variables that may influence behavioral outcomes for simultaneous and sequential bilinguals (such as variables discussed before, including age of onset, proficiency, and frequency of input).

Conclusion

This study used a connectionist self-organizing model to simulate object naming patterns in bilinguals and to identify mechanisms of lexical semantic convergence. We successfully replicated the lexical convergence patterns reported in empirical data from Ameel et al. (2005), and we further investigated the mechanisms and important factors that modulate bilinguals' naming categorization. We demonstrated that the lateral connections play an important role in lexical convergence. Finally, we have identified the role of name agreement strength on bilinguals' object naming. This study provides a first computational model that examines the dynamic interaction between two lexicons in the process of naming objects using single versus multiple languages.

Acknowledgments

This research was supported by a grant from the National Science Foundation (BCS-1057885) to BCM and PL.

References

Ameel, E., Malt, B. C., Storms, G., & Van Assche, F. (2009).Semantic convergence in the bilingual lexicon. *Journal of Memory and Language*, 60(2), 270–290.

Ameel, E., Storms, G., Malt, B. C., &Sloman, S. A. (2005).How bilinguals solve the naming problem. *Journal of Memory and Language*, 53, 60–80.

Kohonen, T. (2001). *The self-organizing maps*. (3rd ed.). Berlin: Springer.

Malt, B. C., Sloman, S. A., Gennari, S., Shi, M., & Wang, Y. (1999). Knowing versus naming: Similarity and the linguistic categorization of artifacts. *Journal of Memory and Language*, 40, 230–262.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Following the Wandering Mind in the Eyes: Tracking Distraction by the Self in a Complex Working Memory Task

Stefan Huijser (s.huijser@rug.nl), Niels Taatgen (n.a.taatgen&rug.nl), Marieke van Vugt (m.k.van.vugt@rug.nl)
Institute of Artificial Intelligence & Cognitive Engineering, University of Groningen, the Netherlands

Keywords: mind-wandering; self-referential processing; eye tracking; cognitive modeling; complex working memory.

Introduction

Do you recognize the experience of being distracted by your own thoughts? Your answer is very likely to be 'yes'. Indeed, a self-report study by Killingsworth and Gilbert (2010) has shown that people spent on average half of their daily activities on thought processes that distract them from their current task. This phenomenon is commonly known as *mind-wandering* and refers to a process of self-generated thought, uncoupled from the external (task) environment (e.g. Smallwood & Schooler, 2015). Although prominent in our lives, cognitive scientists have long ignored it as a subject of study. Until recently, mind-wandering was viewed as a contamination of data and has only been considered as noise in cognitive models (see Vandekerckhove & Tuerlinckx, 2007). This negative view, however, is not entirely ungrounded. According to Smallwood (2015), there are three core challenges that complicate investigating mind-wandering. First of all, it is difficult to experimentally induce due to its, oftentimes, spontaneous occurrence. Moreover, when you do have a working manipulation, the occurrence of mind-wandering is hard to track as it produces little observable overt behavior. Lastly, because it is difficult to measure, self-report is oftentimes resorted to as the only 'valid' measuring technique. However, with mental states such as mind-wandering, there is always a risk that the introspection involved in self-report changes it as well, effectively reducing the validity of this introspection measure (Schooler, 2002). From these three core challenges, we conclude that there is demand for valid experimental manipulations to induce mind-wandering, and for objective measures to capture it. In this paper, we introduce a novel experimental method to track mind-wandering, which can help to validate the specific predictions of a PRIMs model of this process.

Related work

A recent study that tackled the aforementioned issues comes from Daamen, van Vugt, and Taatgen (in press). In this study, an adapted version of a verbal complex working memory (CWM) task was designed to measure and induce mind-wandering. Although many variations of CWM tasks exist, all share the key characteristic that a to-be-remembered item (e.g., a letter) is interleaved by a processing task (e.g., is 'BLICK' a word yes/no) that makes it harder to memorize the items. What is interesting about this study is that a novel manipulation for mind-wandering was introduced: a self-referential processing (SRP) task. SRP refers to a process of thinking about the self (Rogers, Kuiper, & Kirker, 1999) and was induced by asking the participant to judge if a presented trait adjective described themselves yes or no. Alongside the SRP condition, a control condition was used in which participants had to decide whether a presented object word fitted in a shoebox. The results of this study showed a decline in performance for the SRP condition compared to the shoebox condition, possibly reflecting that mind-wandering caused by self-referential processing interfered with rehearsal of the items. This idea was formalized in a cognitive model using the primitive elements model of skill (PRIMs; Taatgen, 2013), which is a symbolic architecture like ACT-R, but is unique in the fact that it supports having multiple competing goals. A key characteristic of the proposed model is that during the processing phase, there is a chance for trait information to remain in working memory where it can initiate productions, or recruit operators in PRIMs terms, that trigger elaborating thoughts on the traits. These 'distraction' operators are modeled to compete with task goal operators to rehearse the items, hence mind-wandering is caused by distraction operators winning the competition. Results showed that the model accounted well for the behavioral data, suggesting that elaborating thoughts on the traits (i.e., mind-wandering) likely interfered with rehearsal and therefore resulted in reduced performance.

The study of Daamen et al. (in press) proposed an interesting and novel way to manipulate and measure the occurrence of mind-wandering. However, there are still some important open issues. First of all, it was only measured indirectly, leaving the question whether the decline in performance in the SRP condition was really due to mind-wandering. Furthermore, although the rehearsal interference mechanism of the model explained the behavioral results well, there was no empirical data to support this mechanism. Therefore, in order for SRP to be confirmed as an objective manipulation for mind-wandering, research is needed to examine if rehearsal interference is an occurring mechanism. Moreover, valid direct measures of mind-wandering need to be identified to follow the occurrence of it across conditions.

Research aims and approach

In the present study, we aim to provide more insight in how mind-wandering can be measured in an objective way.

Specifically, we will attempt to track rehearsal in a CWM task similar to the one employed by Daamen et al. (in press). We test the model prediction that as participants mind-wander, rehearsal will be absent.

Study approach

Behavioral To be able to track rehearsal, we will examine eye movements in a spatial variant of the CWM task. In this task, participants have to memorize the locations of targets (an 'X') in a 4x4 matrix interleaved with a processing sub-task. On every trial, a storage target will be presented first for 1 second. This allows participants to encode the targets' position and to perform rehearsal. Thereafter, a 4 seconds self-paced processing phase will start. Similar to the study of Daamen et al. (in press), this phase will include a SRP and shoebox condition, with the main difference being that the processing words will be presented on the same matrix as the targets. The locations of the words will be random and change every second from position to ensure interference with rehearsal during the processing task itself (thereby restricting it to the 2-s blank periods immediately after the processing phase, which is the rehearsal period according to our model).

The storage phase, processing phase, and blank will be repeated a number of times equal to the span (three and four in this experiment). Thereafter the trial ends with a prompt to recall the storage target locations in the order of presentation.

Thought probes This study will also use thought probes, which are self-report questions aimed at assessing current thought content at various moments in the task. The thought probes will be presented at equally but randomly distributed moments throughout the experiment (48 in total, 9 every block). Although we mentioned that self-report has issues regarding validity, we have chosen to include them alongside eye movement and pupil size, to have a second measure of mind-wandering as control (see section *Eye Tracking* below). Combining both self-report and physiological measures will allow us to give an account on mind-wandering without being tied to one measuring technique.

In this study, we will use an adapted version of the probe question used by Unsworth and Robison (2016). The question is, 'What were you thinking about before you were prompted to answer?', with the following response options: (1) I tried to remember the location of the X's'; (2) I was still thinking about the words from the decision task; (3) I was evaluating aspects of the task (e.g. my performance, how long it takes, difficulty task), (4) I was distracted by my environment (sound/ temperature etc.) or by my physical state (hungry/thirsty); (5) I was daydreaming/ I thought about task-unrelated things, (6) I wasn't paying attention, but I didn't think about anything specifically. Response options 2 to 6 will be counted as attentional lapses, with options 2 and 5 as indicators of mind-wandering.

Eye Tracking The main advantage of using a spatial version of the CWM task is that rehearsal is shown in overt behavior. Memory researchers have found that rehearsal of spatial locations is accompanied with eye movements to these locations (see e.g. Logie, 1995), and that the eye movements can be measured with an eye tracker (Tremblay, Saint-Aubin, & Jalbert, 2006). In this study, eye movements in accordance with the location of previous target locations would be indicative of an active effort to rehearse the target locations from memory. On the other hand, random or absent eye movements would imply that rehearsal is not performed, but that other, possibly goal-irrelevant, processes interfere with rehearsal. Mind-wandering can also be inferred from patterns in the pupil dilation (PD), which is claimed to reflect changes in the attentional state through an indirect link with the norepinephrine system of the locus coeruleus (LC-NE; see e.g, Aston-Jones & Cohen, 2005). Research from e.g. Unsworth and Robison (2016) has indicated that off-task thinking is correlated with low pre-stimulus baseline PD, reflecting a state of relative low arousal and alertness. Moreover, evoked increase in PD due to stimulus processing was found to be lower, further supporting this claim. In this study, we therefore expect that the baseline PD during blanks will be lower on SRP trials and that evoked PD will also be lower during storage. Novel in this study will be that such patterns in PD will be used to predict if participants reported either being on-task (option 1 on thought probe) or off-task (options 2-6).

Conclusion

We have previously shown that a SRP manipulation can impair performance on a CWM task. We have modeled this effect as arising from mind-wandering instigated by the SRP words, which prevents rehearsal. Here we presented a method to test this model by tracking participants' eye movements and pupil size during a spatial analogue of Daamen's CWM task. When successful, this allows us to track mind-wandering by following the eyes.

References

Aston-Jones, G., & Cohen, J. D. (2005). An Integrative Theory of Locus Coeruleus-Norepinephrine Funtion: Adaptive Gain and Optimal Performance. *Annual Review of Neuroscience*, *28*, 403–450.

Daamen, J., van Vugt, M. K., & Taatgen, N. A. (in press). Measuring and modeling distraction by self-referential processing in a complex working memory span task. In *Proceedings 38th Annual Meeting of the Cognitive Science Society*. Philadelphia, PA.

Killingsworth, M. A., & Gilbert, D. T. (2010). A wandering mind is an unhappy mind. *Science*, *330*(6006), 932–932.

Logie, R. H. (1995). *Visuo-spatial working memory*. Hillsdale, NJ: Erlbaum.

Rogers, T. B., Kuiper, N. A., & Kirker, W. S. (1999). Self-reference and the encoding of personal information. In R. F. Baumeister (Ed.), *The self in social psychology* (pp. 139–149). New York, NY: Psychology Press.

Schooler, J. W. (2002). Re-representing consciousness:

Dissociations between experience and meta-consciousness. *Trends in Cognitive Sciences*, *6*(8), 339–344.

Smallwood, J., & Schooler, J. W. (2015). The Science of Mind Wandering: Empirically Navigating the Stream of Consciousness. *Annual Review of Psychology*, *66*(1), 487–518.

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological Review*, *120*(3), 439–471.

Tremblay, S., Saint-Aubin, J., & Jalbert, A. (2006). Rehearsal in serial memory for visual-spatial information: evidence from eye movements. *Psychonomic Bulletin & Review*, *13*(3), 452–457.

Unsworth, N., & Robison, M. K. (2016). Pupillary correlates of lapses of sustained attention. *Cognitive, Affective, & Behavioral Neuroscience*.

Vandekerckhove, J., & Tuerlinckx, F. (2007). Fitting the Ratcliff diffusion model to experimental data. *Psychonomic Bulletin & Review*, *14*(6), 1011–1026.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Towards the Evaluation of Cognitive Models using Anytime Intelligence Tests

Marc Halbrügge (marc.halbruegge@tu-berlin.de)
Quality & Usability Lab, Telekom Innovation Laboratories
Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587 Berlin

Abstract

Cognitive models are usually evaluated based on their fit to empirical data. Artificial intelligence (AI) systems on the other hand are mainly evaluated based on their performance. Within the field of artificial general intelligence (AGI) research, a new type of performance measure for AGI systems has recently been proposed that tries to cover both humans and artificial systems: Anytime Intelligence Tests (AIT; Hernández-Orallo & Dowe, 2010). This paper explores the viability of the AIT formalism for the evaluation of cognitive models based on data from the ICCM 2009 "Dynamic Stocks and Flows" modeling challenge.

Keywords: Anytime Intelligence Test; Model Evaluation; Decision Making; Stock-Flow Problems;

Introduction

Cognitive modeling as a field, although being rooted in AI, has diverged from AI research in recent years because both fields pursue different goals. While modelers try to understand human behavior by creating systems that act as human-like as possible, AI researchers strive for systems that act as perfect as possible, or in Legg and Hutter (2007)'s words: *universal intelligence*. At the same time, parts of the cognitive modeling community are suggesting to direct the field towards more generic models of human behavior ("cognitive supermodels"; Salvucci, 2010) as opposed to task-specific models. Such supermodels did not come into existence yet, but given the methodological advances in the AI field, it may be worthwhile to think about how the abilities (i.e., intelligence) of generic cognitive models should be evaluated. A promising approach to this question are *anytime intelligence tests* (AIT; Hernández-Orallo & Dowe, 2010).

Anytime Intelligence Tests

These intelligence tests are crossing the boundaries between the modeling and the AI field because they are targeting both biological and artificial systems. Based on the work of Legg and Hutter (2007), they intend to measure intelligence of an agent (i.e., 'model' in the terms of the 'other' field) as the accumulated amount of reward[1] r it receives through interaction with a set of (deterministic) environments of varying (computational) complexity. The validity of this accumulated reward is achieved through several means: a) The reward is bound to the range [-1;1]; b) All environments must be balanced, i.e., a random agent will on average receive a reward of zero; and c) The aggregated reward is scaled by the computational complexity of the transition function of the environment. An example of such an intelligence test that was applied to both humans and AI agents can be found in Insa-Cabrera, Dowe, España-Cubillo, Hernández-Lloreda, and Hernández-Orallo (2011).

Besides the construction of new environments that follow the AIT formalism, one can try to analyze published data and models from the literature. This way, potential insights from the AIT procedure can be compared to fit-based evaluations that have been performed before. It is often possible to transform existing tasks into AIT environments by constructing new reward functions for the tasks.

Dynamic Stock and Flow Task

A promising candidate for such a reward reconstruction is the Dynamic Stock and Flow task (DSF; Dutt & Gonzalez, 2007) that has been used for the modeling challenge of the same name (Lebiere, Gonzalez, & Warwick, 2009). The task for this challenge was to maintain the level (i.e., stock) in a water tank at a given target value in the presence of dynamically changing water in- or outflow from an external source. There were four training conditions with monotonously changing inflow (Lin-, Lin+, NonL-, NonL+) and five transfer conditions. Two of these featured linearly increasing inflow (like Lin+), but the agents' actions were delayed by one (Del2) or two (Del3) additional time steps. The remaining conditions featured two (Seq2, Seq2Nos) or four (Seq4) time steps long repeating sequences of inflow; in case of Seq2Nos the pattern was masked by additional noise. The evaluation of the models in the competition was based on the goodness-of-fit to human data in all nine conditions. The crucial variable for this fit was the time-dependent water level.

Besides convenience (i.e., availability of data and models), the DSF task is especially suited as an AIT because it is deterministic and open-ended (in contrast to, e.g., robotic soccer), and the computational complexity of the environment should be both easily scalable and easily quantifiable. Whether and how this task can be transformed to an AIT will now be reviewed regarding possible reward functions for the task. The question of the complexity of the different task conditions will be discussed in a later contribution.

Possible Reward Functions

In the following, r_t denotes the reward for time step t, $amount_t$ denotes the water level for t, env_t the external inflow for t, and *goal* denotes the target water level.

[1] Note: In reinforcement learning, reward functions are a crucial part of the learning agents themselves (Singh, Lewis, & Barto, 2009). In the context of this paper, rewards come from an external 'critic' and were not available to the agents (i.e., human participants and cognitive models) during exploration and learning.

Scaled Absolute Difference. For every time step, the absolute difference to the target water level is multiplied with some constant c and mapped to the range from -1 to 1.

$$r_t = \max(-1, 1 - c|amount_t - goal|)$$

This is the most straightforward solution with the highest face-validity. The agent's proximity to the target level is represented very well. On the other hand, c is arbitrary. Most problematic is that the function is not balanced. Because all task conditions feature external water inflow, random agents would receive an accumulated reward close to the lower boundary of -1.

Relative Progress. A balanced environment could be created by concentrating on the relative progress to the target level. The most simple option is a binary decision whether the water level has improved.

$$r_t = \begin{cases} 1 & \text{if } |amount_t - goal| \leq |amount_{t-1} - goal| \\ -1 & \text{otherwise} \end{cases}$$

This solution has the downside of the current water level being underrepresented. Getting from anywhere to the exact target level would be as good as getting an arbitrarily small amount closer to it. At the same time, environments with external in- or outflow would still not result in random agents receiving a reward of zero.

Relative Progress with Weighting. This can be solved using the following rationale: A perfect agent would always bring the water level to the goal amount in the next step. If this maps to a reward of 1 and no action maps to 0, then the agent should be awarded a reward that is proportional to the stock change made by the agent compared to the two extremes. The result is clipped at -1 in order to stick to the properties of AIT.

$$r_t = \max(-1, \frac{|amount_t - goal|}{|amount_{t-1} + env_t - goal|})$$

Boxplots of the human data collected for the DSF challenge recoded using the proposed reward functions together with the results achieved by a random agent ($flow \sim \mathrm{N}(0,5)$), a null agent ($flow = 0$), and an ACT-R model[2] (Halbrügge, 2010) are given in Figure 1. Of the three proposed reward functions, 'weighted relative progress' provides the best fit to the AIT requirement of balanced rewards.

Discussion and Conclusions

The accumulated reward for the human sample provides interesting evidence about the different difficulty of the nine task conditions. While the four monotonous conditions on the left (Lin- to NonL+) are all comparatively easy, the delay conditions are very hard. In the Del3 condition, the median of the human sample is close to random performance. The difficulty of the sequence conditions lies between the monotonous and the delay conditions.

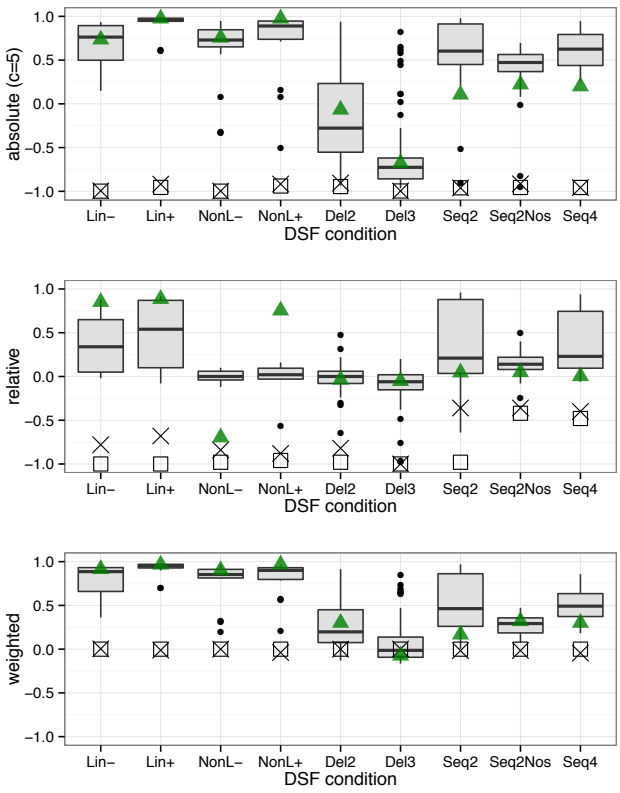


Figure 1: Average reward in the DSF task after recoding using the three proposed reward functions. Boxplots: Human Data. Squares: Null Agent (no action). Crosses: Random Agent. Triangles: ACT-R model (Halbrügge, 2010).

The performance of both humans and the model will be compared to the computational complexity of the respective environments. Then, the models that had entered the competition can be evaluated with respect to their intelligence as opposed to their fit to the human data.[3] Such an evaluation should consider the computational complexity of the models as well (Halbrügge, 2007). Complexity metrics based on the source code could be accompanied by Model Flexibility Analysis (Veksler, Myers, & Gluck, 2015), which tries to estimate the range of possible model behavior through simulation (see also Gluck, Stanley, Moore, Reitter, & Halbrügge, 2010). Together with the AIT formalism, this could lead to a new evaluation criterion that would be complementary to fit measures like R^2 and RMSE and could also provide a step towards reuniting the fields of cognitive modeling and artificial (general) intelligence.

[2]The source code of the cognitive model is available for download at http://dx.doi.org/10.14279/depositonce-5163

[3]Especially the delay conditions often lead to oscillating stock levels, which renders averaging across participants questionable.

References

Dutt, V., & Gonzalez, C. (2007). Slope of inflow impacts dynamic decision making. In *Proceedings of the 25th international conference of the system dynamics society* (p. 79). Boston, MA.

Gluck, K. A., Stanley, C. T., Moore, L. R., Reitter, D., & Halbrügge, M. (2010). Exploration for understanding in cognitive modeling. *Journal of Artificial General Intelligence*, *2*(2), 88–107. doi: 10.2478/v10229-011-0011-7

Halbrügge, M. (2007). Evaluating cognitive models and architectures. In G. A. Kaminka & C. R. Burghart (Eds.), *Evaluating architectures for intelligence. papers from the 2007 aaai workshop* (p. 27-31). Menlo Park, California: AAAI Press.

Halbrügge, M. (2010). Keep it simple – a case study of model development in the context of the dynamic stocks and flows (dsf) task. *Journal of Artificial General Intelligence*, *2*(2), 38–51. doi: 10.2478/v10229-011-0008-2

Hernández-Orallo, J., & Dowe, D. L. (2010). Measuring universal intelligence: Towards an anytime intelligence test. *Artificial Intelligence*, *174*(18), 1508–1539. doi: 10.1016/j.artint.2010.09.006

Insa-Cabrera, J., Dowe, D. L., España-Cubillo, S., Hernández-Lloreda, M. V., & Hernández-Orallo, J. (2011). Comparing humans and ai agents. In *Artificial general intelligence* (pp. 122–132). Springer. doi: 10.1007/978-3-642-22887-2_13

Lebiere, C., Gonzalez, C., & Warwick, W. (2009). A comparative approach to understanding general intelligence: Predicting cognitive performance in an open-ended dynamic task. In *Proceedings of the second conference on artificial general intelligence* (pp. 103–107). Amsterdam: Atlantis Press.

Legg, S., & Hutter, M. (2007). Universal intelligence: A definition of machine intelligence. *Minds and Machines*, *17*(4), 391–444. doi: 10.1007/s11023-007-9079-x

Salvucci, D. D. (2010). *Cognitive supermodels.* (Invited talk, European ACT-R Workshop, Groningen, The Netherlands)

Singh, S., Lewis, R. L., & Barto, A. G. (2009). Where do rewards come from? In *Proceedings of the annual conference of the cognitive science society* (pp. 2601–2606).

Veksler, V. D., Myers, C. W., & Gluck, K. A. (2015). Model flexibility analysis. *Psychological Review*, *122*, 755-769. doi: 10.1037/a0039657

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Analyzing fatigue, stress and human errors in Emergency Operation Centre management: The consequences of using different cognitive modelling frameworks

Robert L. West (robert.west@carleton.ca), Korey MacDougal (koreymacdougall@gmail.com)
Institute of Cognitive Science, Carleton University, Ottawa, Ontario, Canada

Lawrence M. Ward (lmward@psych.ubc.ca)
Department of Psychology, University of British Columbia, Vancouver, British Columbia, Canada

Keywords: Cognitive Modelling; ACT-R; Cognitive Architecture; Macro Cognition; Emergency Operations Centre; Stress; Fatigue.

Emergency Operation Centres (EOCs) are responsible for planning and responding to large scale disasters, such as Katrina, Fukushima, or any large earthquake. An EOC is a central command for coordinating the different field assets (e.g., fire, police, ambulance, social services). It is usually composed of several individuals (EOC managers), each of whom has a designated set of tasks, such as communication, logistical support, mapping events, writing reports, as well as a manager who is in charge of coordinating everything. EOC mangers normally do not directly control field assets; instead the managers pass on information and make strategic recommendations.

EOC organization and training is based on guidelines issued by government bodies, such as FEMA in the US. However, EOC performance is very difficult to study because massive disasters are infrequent and there is very little data available afterwards. Also, it is highly problematic and expensive to realistically simulate disasters in the lab. Cognitive modelling provides a pragmatic avenue for addressing this issue in lieu of directly studying EOC managers under realistic conditions.

However, all cognitive modelling is not the same and different cognitive modelling frameworks are likely to generate different types of recommendations. Because of this it is important to consider the range of cognitive modelling frameworks at the outset of a project, to avoid using a single framework that might generate poor or dangerous recommendations. Different cognitive modelling frameworks can be treated as different ontological and/or epistemological systems (i.e., what is the nature of the object of study and how best to understand the object of study, respectively). By conceptually analyzing how each framework applies to the project a better sense of the project as a whole can be developed and cognitive modelling can be deployed more effectively.

In this short paper we have applied this methodology to understanding how to tackle the problem of fatigue in EOCs. One of the main principles of EOC management is to protect the first responders (e.g., firemen, police, ambulance). So, if a building has collapsed and is unstable, first responders should not be sent in if there is a chance of further collapse, even if this means that those needing immediate help will die. The reasoning behind this is that if the first responders die then there will be no one to help the others. This reasoning can be extended to fatigue and stress within the EOC. That is, although they are not in physical danger, EOC managers are in danger of mental fatigue, which could lead to serious mistakes or misjudgments.

General Guidlines

One solution that has been proposed is to use models of fatigue to generate generalized guidelines for breaks and then enforce the breaks. This would involve having a mental health professional, or someone trained to monitor for fatigue on the team, and giving them the authority to enforce breaks.

Cost/Benefit

Using enforced breaks could reduce fatigue but, even if we assume mental health professionals can effectively discern fatigue *and* that the other EOC workers will obey them, there is still an issue. The model should be framed in terms of a cost benefit analysis, where the benefit is avoiding errors caused by cognitive fatigue and the cost is the information lost when a manager is replaced with another manager. That is, loss of information can also result in serious errors.

Human Factors/HCI

A major component of EOC management is logging the ongoing flow of information. In theory this should mitigate the problem of information loss when an EOC manager needs to rest. Information logging is an area where Human Factors studies and Human Computer Interaction evaluations can be used to develop more efficient systems for logging information. This would both improve information transfer and reduce fatigue due to poorly designed systems.

However, focusing on individual systems instead of looking at the whole picture can lead to premature optimization (Knuth, 1974). That is, improving the efficiency of the parts may only produce small, insignificant improvements overall, and it could even make the whole system worse (Gray et al, 1993). In the case of the EOC, although logging information is an important part of EOC management, the purpose of the EOC is to integrate information and maintain a functional awareness of the overall situation. This is not something that cannot be logged in the same way as specific events. Therefore, improving the efficiency of logging specific events needs to be done within the context of maintaining a common ground (Klein et al, 2004) situational awareness.

Unit Tasks

Cognitive modelling methodology often proceeds by first identifying the unit task structure and then modelling the individual unit tasks. Unit tasks are theorized to divide up a task into parts, such that the cognitive system is not overloaded and down time is avoided (Card et al, 1983). Specifically, unit tasks are designed so that all of the information needed can be processed by the cognitive system in real time (avoid overload), and so that unit tasks do not get hung up waiting for something to happen when the agent could be getting something else done (avoid downtime). More recently, it has been proposed that unit tasks are also designed to avoid interruptions. That is, a unit task will be of a size such that it is likely that it will be finished without interruption (West & Nagy, 2007).

If we take the unit task concept seriously, it provides an important insight. Specifically, if EOC managers consider resting as downtime, and not an integrated part of the management task, then they will tend to have unit task structures that avoid it. This makes sense as many EOC managers are drawn from police, fire, or ambulance services where it is unusual to have emergencies that last more than a few hours, so they can normally rest after completing their tasks. Under these conditions, treating rest as downtime and minimizing it makes sense. These professionals are often chosen as EOC managers because EOCs need people who are fast and efficient when required. So anything that interferes with that could be problematic. Importantly, this may include enforced breaks and unfamiliar logging methods.

Macro Cognition

Macro cognition can involve a number of different methods and theoretical approaches, however, we will focus on the SGOMS modelling framework, as it seems particularly useful for understanding EOC management. A planning unit is an SGOMS structure that serves a similar purpose as the unit task. Whereas, unit tasks are a control structure (Card et al, 1983) for protecting the integrity and efficiency of the *micro* cognitive architecture (e.g., memory, attention, motor actions, perception, etc.) planning units are a control structure (West & MacDougal, 2015) for protecting the integrity and efficiency of the *macro* cognitive architecture (e.g., planning, cooperation, interruptions, reacting to unexpected events, etc.). Planning units can be thought of as a way of managing unit tasks in that the appropriate unit tasks need to be completed to complete a planning unit. In this sense, planning units control the flow of unit tasks. However, unlike unit tasks, planning units are designed to be interrupted and restarted. Importantly, planning unit choice is based on context and situation awareness.

From this perspective, the problem is that EOC managers do not have a rest-break planning unit. As noted above, there is no reason why they would since extended rest-breaks are not part of the normal routine in police, fire, or ambulance services. Resting should be a specific planning unit, just as logging information would be a specific planning unit. A model of how best to do this could be used as the basis for designing a training program. A resting planning unit would be triggered by downtime and would contain relevant unit tasks such as: locate a place to rest, inform colleagues about rest, arrange for someone to cover your post, arrange for when your rest should end, and pass on any important information.

Conclusions

We have analyzed the problem of fatigue in EOC workers using various different modelling frameworks as ontological and epistemological tools. Each solution seems reasonable when viewed in isolation but, in fact, they may produce solutions that are problematic. Using a model to say when a rest break should be enforced is a responsible and principled way to implement this policy, but this policy could lead to serious problems if information transfer is ignored. Using Human Factors and HCI to improve information logging could ameliorate this, but these solutions need to be evaluated in the broader context of maintaining a common ground situational awareness. Applying the unit task concept shows why EOC operators avoid downtime but also demonstrates that this is a necessary consequence of having fast, efficient responses. One consequence of this is that care needs to be taken that an isolated solution or improvement doesn't lead to a less efficient overall system (e.g., as in Gray et al, 1993). Finally, the planning unit concept suggests that the focus should be first on training rather than enforcement or systems efficiency.

The more general point we are making is that applied cognitive modelling should involve an initial assessment of the whole task to get a broad understanding of how modelling can be best applied. As we have shown, this can be done by applying concepts drawn from the field of cognitive modelling itself.

References

Card, S. K., Newell, A., & Moran. T. P. (1983). The Psychology of Human-Computer Interaction. L. Erlbaum Associates Inc. Hillsdale, NJ.

Gray, W. D., John, B. E., & Atwood, M. E. (1993). Project Ernestine: Validating GOMS for predicting and explaining real-world task performance. *Human Computer Interaction*., *8*(3), 237-309.

Klein, G., Feltovich, P. J., Bradshaw, J. M., & Woods, D. D. (2004). Common ground and coordination in joint activity. In W. B. Rouse & K. R. Boff (Ed.), Organizational Simulation. (pp. 139-184). New York City, NY: John Wiley.

Knuth, D. (December 1974). Structured Programming with Goto Statements", *Computing Surveys* **6**:4, pp. 261–301

West, R. L., & Nagy, G. (2007). Using GOMS for Modeling Routine Tasks Within Complex Sociotechnical Systems:Connecting Macrocognitive Models to Microcognition. Journal of Cognitive Engineering and Decision Making

West, R. L., & MacDougal, K. (2015). The Macro Architecture Hypothesis: Modifying Newell's System Levels to Include Macro Cognition. *Biologically Inspired Cognitive Architectures*.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Connecting Cognitive Models to Interact with Human-Computer Interfaces

Farnaz Tehranchi (fjt5064@psu.edu)
Department of Computer Science and Engineering
Penn State, University Park, PA 16802 USA

Frank E. Ritter (frank.ritter@psu.edu)
College of Information Sciences and Technology
Penn State, University Park, PA 16802 USA

Keywords: Cognitive Model; Cognitive Architecture; Human Computer Interface, Interaction.

Introduction

Cognitive modeling goals include understanding and predicting human behavior. In fact, the main objective of cognitive science is understanding the nature of the human mind to develop a model that predicts and explains human behavior. Cognitive architectures are infrastructures for cognitive science theory and provide computational frameworks to execute theories.

ACT-R is an architecture of cognition (Anderson, 2007): a platform that is used to implement cognitive models. ACT-R communicates with the outside world by using modules such as visual perception and motor control. In this paper, we propose a model that will provide an environment for ACT-R to interact with the world. For this purpose, we use EMACS as an interactive text editor that includes extensions to read email, browse the web, and work with spreadsheets (Ritter & Wood, 2005). Both ACT-R and Emacs are written in Lisp. It allows us to extend them and design a bridge between them. This bridge will enable ACT-R to communicate with an interactive environment. Further steps within this work involve expanding the model to be more human-like.

The two main needs for a cognitive model to interact with a task environment are 1) the ability to pass commands, and 2) the ability to access the information on the screen. There are two main approaches for interaction. The first approach to meeting these needs is to use a graphic language library such as MCL, Tcl/Tk, Java, or SL-GMS to define objects and pass them from an interface to simulated hands and eyes (Ritter, Baxter, Jones, & Young, 2000) such as ACT-R/PM (Byrne & Anderson, 1998). The second approach is using the screen's bitmap and transforming the image into objects and symbols that a cognitive model can manipulate. For example, SegMan (St. Amant, Riedel, Ritter, & Reifers, 2005) provides domains in which SegMan applies to support cognitive model-based interaction and evaluation. However, SegMan is not yet user-friendly and easy to extend. It still does not recognize everything on the display.

In addition, there are several other approaches for providing models access to tasks, such as simulating the task in the models' heads, being passed a list of inputs from a program that reads from a file of input sets, and instrumenting a particular task interface.

In this report, we extend a previous attempt to provide ACT-R access to the world (Kim, Ritter, & Koubek, 2006) by enabling ACT-R to interact with the Emacs text editor. In this work, ACT-R can communicate with a task environment by instrumenting a graphical library, in this case the Emacs text editor.

The ESegman Approach

ACT-R has been introduced by Anderson (1993) for implementing cognitive models. In fact, it is an architecture of cognition that makes precise predictions about behaviors. Any ACT-R cognitive model has a loop with three components: the task environment, the model, and the results (see Figure 1). The cognitive model receives its perspective as input from the visual (and auditory) module and outputs actions through the motor module. The cognitive architecture within this process understands the external world this way.

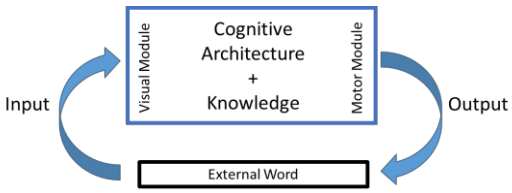


Figure 1: Cognitive model structure.

Figure 2 shows that the first step in the modeling process could be collecting data from the real world (although some would argue for and start at block 3). This process can be performed using the Dismal spreadsheet task. Humans have directly performed the Dismal spreadsheet tasks (Kim & Ritter, 2015), and their performance is recorded using the Recording User Input (RUI) software (Kukreja, Stevenson, & Ritter, 2006; Morgan, Cheng, Pike, & Ritter, 2013).

In particular, for our connection between the architecture and the world, we will use Emacs functions to take the user commands from the main process (ACT-R will be a sub-process of the Emacs process) and insert them into the target buffer. Therefore, the model will be aware of the user commands and can execute them. Whether it is a keystroke command or a mouse action, the model will be capable of using the Emacs Lisp language to execute them. After collecting the commands from ACT-R and receiving requests to 'look' and move the eye, we will feed ACT-R

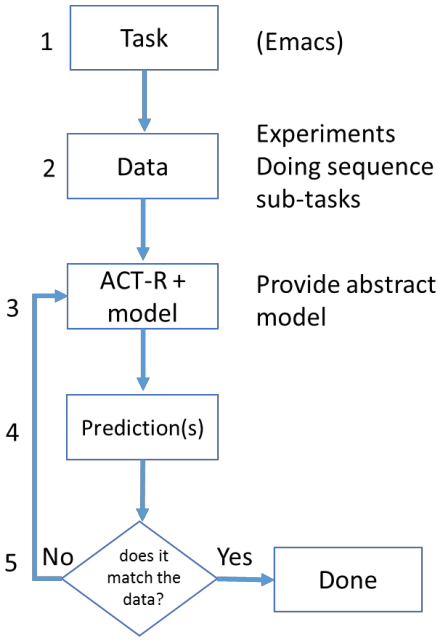


Figure 2: Model process.

with the information about the world by inserting target buffer contexts into its visual module. This implements parts of block 3 in Figure 2 and the connection in Figure 1.

We will also be able to use an existing ACT-R model to interact with Emacs. Paik et al. (2015) created this set of ACT-R models to perform the Dismal spreadsheet task. These models generate the actions to perform the task, but the actions are not implemented in the world—they are simply generated. Enacting them in the world (Emacs in this case) would help test the timing, and with the actions passed in the world, one could turn on errors in the perceptual-motor component in ACT-R and start to explore how errors are generated, noticed, and recovered from.

The main challenge will be the communication step. In particular, ACT-R can load data into Lisp. For sending and receiving data it needs to communicate through the main Lisp application process with Emacs.

In our approach, we are going to load ACT-R within a buffer managed by SLIME, which is an Emacs mode for Common Lisp development. Our objectives will be connecting the ACT-R process with the Emacs that is already managing ACT-R. One of the potentials of Emacs would be allowing an ACT-R model to edit files in the Emacs environment that it is running in.

Another method would be sending a request to an already running Emacs that is waiting for a socket request (using the Swank library), and setting its output as one of the parameters of make-network-process function for opening a client TCP/IP connection.

From a security perspective, we should consider a Swank address that is not wide open to the world because it can make the computer vulnerable to intrusion attacks. However, in our connection scenario, we will not access Swank from another computer. Therefore, establishing a secure channel to the Swank server will not be necessary.

Conclusion and Further Research

Cognitive models have struggled connecting to the world. This approach of connecting through and with Emacs will provide another opportunity for providing models access to more interesting tasks.

Acknowledgments

This work was funded by ONR (N00014-11-1-0275 & N00014-15-1-2275). David Reitter has provided useful comments on Emacs and Aquamacs, (the Emacs version for the Mac). Jong Kim provided the idea.

References

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?* New York, NY: Oxford University Press.

Byrne, M. D., & Anderson, J. R. (1998). Perception and action. In J. R. Anderson & C. Lebiere (Eds.), *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Kim, J., Ritter, F. E., & Koubek, R. J. (2006). ESEGMAN: A substrate for ACT-R architecture and an Emacs Lisp application. In *Proceedings of ICCM - 2006- Seventh International Conference on Cognitive Modeling*, 375. Edizioni Goliardiche: Trieste, Italy.

Kim, J. W., & Ritter, F. E. (2015). Learning, forgetting, and relearning for keystroke- and mouse-driven tasks: Relearning is important. *Human-Computer Interaction, 30*(1), 1-33.

Kukreja, U., Stevenson, W. E., & Ritter, F. E. (2006). RUI—Recording User Input from interfaces under Windows and Mac OS X. *Behavior Research Methods, 38*(4), 656–659.

Morgan, J. H., Cheng, C.-Y., Pike, C., & Ritter, F. E. (2013). A design, tests, and considerations for improving keystroke and mouse loggers. *Interacting with Computers, 25*(3), 242-258.

Paik, J., Kim, J. W., Ritter, F. E., & Reitter, D. (2015). Predicting user performance and learning in human-computer interaction with the Herbal compiler *ACM Transactions on Computer-Human Interaction, 22*(5), Article No.: 25.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2000). Supporting cognitive models as users. *ACM Transactions on Computer-Human Interaction, 7*(2), 141-173.

Ritter, F. E., & Wood, A. B. (2005). Dismal: A spreadsheet for sequential data analysis and HCI experimentation. *Behavior Research Methods, 37*(1), 71-81.

St. Amant, R., Riedel, M. O., Ritter, F. E., & Reifers, A. (2005). Image processing in cognitive models with SegMan. In *Proceedings of HCI International '05*,Volume 4 - Theories Models and Processes in HCI. Paper # 1869. Erlbaum: Mahwah, NJ.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

An Update on Automatic Transcription vs. Manual Transcription

Frank E. Ritter (frank.ritter@psu.edu)
Catherine Bouyat (cmb6187@gmail.com)
Kaitlyn Ekdahl (kaitlyn.ekdahl@gmail.com)
Daniel Guzek (dan.guzek@psu.edu)
College of Information Sciences and Technology, Penn State University
University Park, PA 16802 USA

Abstract

Cognitive modelers have long used verbal protocol analysis to gather data to test their models. Recently developed tools offer support for automatic transcription of audio. In this short paper, we compare the time it takes to transcribe a video done (a) exclusively with Google's subtitle tool, (b) corrected from Google's subtitle tool, and (c) done completely by hand. We found that using the subtitle tool alone can yield too high an error rate, correcting Google subtitles took about 2.5x the video length, and transcribing completely by hand took approximately 11x the video length. We can thus recommend using Google subtitles as a starting point for verbal transcription as it offers a useful speed up in transcription. In addition, we can recommend when and how to use Google subtitles, and the use of headphones and automatic spelling correction in text editors.

Keywords: Transcription, Google closed captioning, YouTube, Verbal protocol analysis

Introduction

Cognitive modelers have long used verbal protocols (Larkin, McDermott, Simon, & Simon, 1980; Newell & Simon, 1972). The transcription of these protocols can be problematic, often taking 10x longer to transcribe than record (Ericsson & Simon, 1993; Ritter & Larkin, 1994). Thus, there is interest in automatic protocol transcription systems.

In this paper we examine a new tool to perform automatic verbal protocol transcription: YouTube's automated closed captioning service. We start to explore how to use the service more efficiently by examining how long it takes to transcribe an example video and note some lessons about how to assist transcription.

Method

In this pilot study, we took an example 47-minute video and used three approaches to transcribe its content: we copied the transcript created by YouTube's closed captioning (CC) service directly from YouTube; we copied YouTube's transcript and had a human coder manually correct it while listening to the video; and we manually transcribed a portion of the video from scratch. The human coder stopped out of frustration after approximately three hours of manual transcription, having successfully transcribed only 15 minutes of content, so we use that amount in our analyses.

Apparatus and Material

We used Google's CC service on YouTube to transcribe a video of a seminar presentation (https://www.youtube.com/watch?v=RcZU-fb0Q10), chosen somewhat arbitrarily as an example of naturalistic, public speech rather than a strictly concurrent verbal task protocol.

The coders each accessed the video on a PC and Mac laptop. They used either a Google Chrome (Mac) or Mozilla Firefox (PC) browser to view the video, and Microsoft Word (PC) or TextEdit (Mac) to hold and edit the transcripts.

Participants/Coders

The two coders were undergraduates in the College of IST at Penn State, whose first language is English. They have taken multiple courses in HCI and worked as research assistants for at least six months.

Design and Procedure

Both coders were each given the link and asked to transcribe the audio. Coder 1 (PC, arbitrarily chosen) worked first with the YouTube CC-generated transcription and edited the transcription as they watched the video. Coder 2 (Mac, arbitrarily chosen), transcribed by hand.

During the transcription process, the coders each used two windows—a browser and a text editor—on their own laptops. Headphones were also used, and both transcriptions were done in quiet spaces.

Results and Discussions

Table 1 shows an example of the automatic transcription for both speakers involved in the example video.

Table 1. Example of the unedited automatic transcription.

Example of unedited transcription from Speaker 1
0:03 ok once we get started my neighbors call for a research professor I have been
0:16 here for various events if you are out for those of you that don't have a lot
Example of unedited transcription from Speaker 2
3:17 this point it becomes how could I ever that confused but I was very green at
3:20 that point and so I'm just gonna talk a little bit about who's who in the zoo is

Table 2 presents the time to transcribe the video, including that the transcription time using the YouTube CC editing is nearly instantaneous and would be constant across length.

The time to correct the video's transcript takes about 4x the video's length. The time to manually transcribe, however, is about 11.4x the video's length, which is consistent with previous estimates.

Table 2. Time to transcribe the first 15 min. of the sample video.

Transcription Method	Time (min.)	Ratio to video length	Ratio to Manual
YouTube CC Transcript	2*	0.14	0.01
YouTube CC Edited Transcript	67	4.4	0.39
Manual Transcription	171**	11.4	1.0

* Does not include time to upload the video (32 min.) on ~6 Mbps link. Time to provide an automatic transcription is variable and may take several hours or a day.
** Only the first 15 min. were transcribed.

The YouTube CC transcript does not include the time needed to correct most grammatical errors (as YouTube's closed captions lack punctuation).

We also noticed that audio quality impacted error rates in the generated transcription. Fixing the generated transcript of the video's first speaker (0:00-2:13) took 27 min. The correction rate of 12.2 min./1 min. of audio is roughly on par with previous rates. This audio portion was less structured, consisted of more informal dialogue, and was articulated less clearly because the podium microphone was farther from the speaker.

Fixing the generated transcript of the second speaker (2:14-15:03) took only 46 min., or 3.6 min./min. This speaker wore a clip microphone, with clearer audio. Additionally, his speech was more practiced and steadily paced, which contributed to more correct grammar in the generated transcription.

Discussion and Conclusions

Based on this process, we can make several suggestions for how to do further verbal protocol transcriptions. (a) It can be useful to run a block of audio through YouTube's closed captioning tool to generate an automatic transcript. This may be possible using YouTube's private settings or using publicly available video. Editing the automatic transcript appears to save a lot of time if the audio is as clear as in our single example.

(b) We recommend using headphones to provide better quality sound. Headphones allow transcribers to better understand the speaker in the video.

(c) YouTube's video settings allow the user to adjust the speed of the video. By adjusting the speed manually, the audio can be slowed down to better recognize the words being spoken.

(d) Modern text editors can assist transcription efforts because they can autocorrect many typos. Coder 2 found that TextEdit was superior to Word and that the tradeoff between corrections vs. over-corrections (or lack of corrections) was worthwhile.

There may be a few limitations to this approach. Auto-correction in text editors could be a drawback in some cases if non-standard speech is being analyzed. Also, the video we analyzed might not be representative of verbal protocols. Future work should test more naturalistic verbal protocol material.

In addition, we can note a second, even more automatic method for obtaining a transcript from videos on YouTube. This method uses a Python-based command-line utility, youtube-dl, which works on Unix and Windows systems as long as a Python interpreter is present. The utility when passed the argument "--write-auto-sub" will download the video file in .mkv format and the automated captions in vtt-format. The .vtt file provides resolution to millisecond precision about when YouTube should highlight each word.

While the first method provided less timestamp information, it sufficed for our purposes. The .vtt file would have needed parsed to extract the transcript and discard all text coloring metadata, a task for which no such tool publicly exists at this time.

It appears now possible to automatically transcribe verbal protocols, at least approximately, using YouTube's closed captioning with a few errors, or with about a 4x cost to correct errors. The ability to use verbal protocols seems to have become easier, particularly where the audio is clear.

Acknowledgments

This work was funded by ONR (N00014-15-1-2275). We thank Ysabelle Coutu for ways to improve this paper.

References

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data* (2nd ed.). Cambridge, MA: MIT Press.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Expert and novice performance in solving physics problems. *Science, 208*, 1335-1342.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Ritter, F. E., & Larkin, J. H. (1994). Developing process models as summaries of HCI action sequences. *Human-Computer Interaction, 9*(3&4), 345-383.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Modeling Human Intention in a Live-Feeling Platform

Martin Lukac(martin.lukac@nu.edu.kz)
Department of Computer Science, Nazarbayev University,
Astana, 010000, Kazakhstan

Gaziza Oteniyaz (goteniyaz@nu.edu.kz)
Department of Computer Science, Nazarbayev University,
Astana, 010000, Kazakhstan

Michitaka Kameyama (michikameyama@isenshu-u.ac.jp)
Graduate School of Information Sciences,
Tohoku University, Sendai, Japan

Abstract

This paper presents a Bayesian Network (BN) model of human intention in an entertainment environment. We present a framework for live-feeling communication that allows an intelligent system to reconfigure its action based on user preferences and intention. The user intentions are learned from a training set of model situations and a hierarchical model of soccer game is integrated with the user intention into a single BN. The trained model is tested on a data set gathered from users. The BN model shows an accuracy of up to 85%.

Keywords: user intention estimation, live-feeling communication, live entertainment.

Introduction

Intention estimation is a crucial part of intelligent systems intended to be used in daily life or solving problems along with humans (Omori et al. 2007, Yokoyama et al. 2008, Kuan et al. 2010). The reason is that in order to make human-machine cooperation seamless and not requiring a constant direct input from human, the machine must be able to predict user's intention within the context of the task.

Watching a live game on site at a stadium allows the spectators to fully enjoy the entertainment feature since spectators can always view and follow what is happening by turning the head to the right direction. Moreover spectators can decide to observe not only the game but particular elements of the game, other spectators or the environment.

In order to allow spectator to fully enjoy a live entertainment event we propose a live-feeling platform that based on user preferences and the real time content of the event will provide the most desirable view to the user. The platform takes as input real time situation, user preferences and outputs set of commands to one of the available camera. The result of the camera commands are evaluated and compared to expected content that the spectator desires to see.

2. Live-Feeling Communication Platform

The base for our research is shown in the conceptual depiction of the live-feeling communication platform shown in Figure 1. The idea is as follows. The user, being at a remote location is observed by the computer and a real time recognition of emotion, as well as user preferences and game statistics are recorded in real time. The Figure 1 illustrates the camera and microphones to collect data of user intention. On the other end a set of mobile camera controlled by a computer are showing a real-tie event such a soccer or baseball.

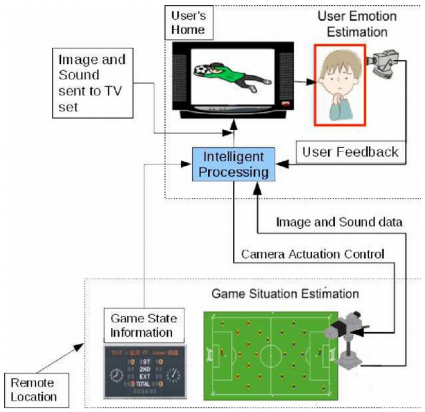


Figure 1: The live-feeling communication platform.

The target of the proposed setup is to show to the user the desired content based on his/her preferences in order to maximize his/her happiness and improve the experience of the event.

3. Data Collection

We extracted the following situations after considering them as the most important moments in a football game: offside, penalty kick, corner kick, goal kick, throw-in, free kick, moment of fouls and misconducts. In order to collect data that is equivalent to sensor processed results, we interviewed people, who are interested in football, for each of these situations. 10 people participated in interviews.

Output discretization

In order to allow predicting the camera control, we divided the football field into the grids and numerated them.

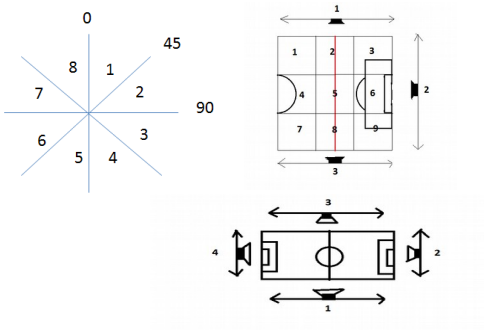


Fig.2 Camera's angles, positions and one half of the football pitch divided into grids

From figure 2 it can seen that in order to show position of some user preference, model specify number of cameras, grids where each should be located and angle. Zoom is specified by 3 levels: short, normal and wide. For example if network gives an output for offside situation "camera #3, grid #8, angle #1, normal zoom" then it means that during offside user may want to see position of goalkeeper since network showing the grid #6.

There are overall 5 cameras. We assume that camera #1, #2, #3, #4 can move, but camera #5 does not, because it is on the top.

4. Model of Human Intention

The Bayesian model of human intention consist of a set of input and output nodes. Inputs of the BN are user preferences and game event situations; outputs are number of camera, camera position, angle and zoom. Inputs node represent categories that were extracted from data analysis.

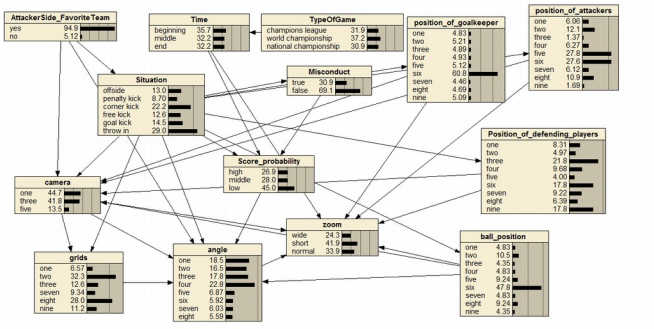


Fig.3 Network for Intention of User Estimation

5. Experiments and Results Discussion

Our goal is to have a human intention model that accurately recognizes human intention and respectively give an output as camera position, its angle and zoom that appropriately capture user preferences.

We conducted interviews to test model accuracy with user cognition. 14 people participated in interviews used for the testing.

In order to calculate accuracy of the network, we compared possible position of user preferences, which we obtained from testing, to position that is indicated by network. Network shows certain field position that we obtained by camera position, its angle and zoom. These places we specified as numerated grids.

Table 2: Network accuracy

Situation	User preferences	Expected locations	Network accuracy
Offside	goalkeeper:20%	Grid #6	20%
	defending players:20%	Grid #6	25%
	referee decision:20%	Grid #7,8,9	10%
	offside position in the pitch:40%	Grid # 6, 5	25%
			Overall:85%

Table 2 illustrates results of network accuracy in the offside situation. After testing network we obtained what user wants to see during offside situation. User preferences of people are goalkeeper, defending players, referee decision and position where offside happened. In the Table 2 it can be seen how many people prefer to see each of this user preferences out of 14 people in percentage. There are also expected locations of each user preferences. The last column illustrates percentage of accuracy out of 25% for each preference. What network shows during offside position is shown in the figure 4. As it can be seen the red square is the position shown by network with wide zoom. After comparison of expected positions with position that is indicated by red square on the figure 4, we identified that network can show goalkeeper, defending players and offside position in the pitch with high accuracy. However network cannot identify position of referee and its results, since referee can be located outside of the pitch near 7, 8, 9 grids. Despite this fact, there is 10% of accuracy for showing referee, since camera #1 can view 9th grid but with little accuracy since zoom is wide and it can clearly show only 6th grid.

6. Conclusion

In this paper we demonstrated a study on intention estimation during a live event in a live-feeling communication platform. The proposed model works well for the statistically gathered information however as of yet lacks the real-time ability to adapt for individual user's emotional expressions.

In the future a more complete model estimating only user intention (what is the desired content instead of camera control) is to be developed. Moreover building a model for individual user intention estimation is also intended as future work.

References

Chalnick, A., & Billman, D. (1988). Unsupervised learning of correlational structure. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society* (pp. 510-516). Hillsdale, NJ: Lawrence Erlbaum Associates.

Feigenbaum, E. A. (1963). The simulation of verbal learning behavior. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought.* New York: McGraw-Hill.

Hill, J. A. C. (1983). A computational model of language acquisition in the two-year old. *Cognition and Brain Theory*, *6*, 287-317.

Kuan, J. Y., Huang, T. H., & Huang, H. P. (2010, August). Human intention estimation method for a new compliant rehabilitation and assistive robot. In *SICE Annual Conference 2010, Proceedings of* (pp. 2348-2353). IEEE.

Ohlsson, S., & Langley, P. (1985). *Identifying solution paths in cognitive diagnosis* (Tech. Rep. CMU-RI-TR-85-2). Pittsburgh, PA: Carnegie Mellon University, The Robotics Institute.

Oliver, N. M., Rosario, B., & Pentland, A. P. (2000). *A bayesian computer vision system for modeling human interactions. Pattern Analysis and Machine Intelligence*, IEEE Transactions on, 22(8), 831-843.

Omori, T., Yokoyama, A., Okada, H., Ishikawa, S., & Nagata, Y. (2007, November). Computational modeling of human-robot interaction based on active intention estimation. In *Neural Information Processing* (pp. 185-192). Springer Berlin Heidelberg.

Lewis, C. (1978). *Production system models of practice effects*. Doctoral dissertation, Department of Psychology, University of Michigan, Ann Arbor.

Newell, A., & Simon, H. A. (1972*). Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Sun, X., Jin, G., Huang, M., & Xu, G. (2003, September). Bayesian-network-based soccer video event detection and retrieval. In Third International Symposium on Multispectral Image Processing and Pattern Recognition (pp. 71-76). International Society for Optics and Photonics.

Shrager, J., & Langley, P. (Eds.) (1990). *Computational models of scientific discovery and theory formation*. San Mateo, CA: Morgan Kaufmann.

Yokoyama, A., Omori, T., Ishikawa, S., & Okada, H. (2008). Modeling of action decision process based on intention estimation. In *SCIS & ISIS* (Vol. 2008, No. 0, pp. 328-333). 日本知能情報ファジィ学会.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

A Bond Graph Approach for Wellness Management based on the Client-Therapist Model

ABDELRHMAN MAHAMADI (aam103@zips.uakron.edu)

SHIVAKUMAR SASTRY (ssastry@uakron.edu)
Department of Electrical and Computer Engineering
The University of Akron, Ohio USA

Abstract

There is an urgent need to represent and reason about human behavior to enable a large number of emerging applications. Our interest is to understand how to encourage behaviors that result in promoting wellness management for individuals. In this short paper, we present a bond graph model for modeling human behavior. We demonstrate that these versatile models are useful to represent energy transfer across multiple domains. The approach offers us a systematic method to analyze the models and derive dynamic equations to represent the behaviors.

Keywords: Bond Graph, Human Behavior.

Introduction

Exploring human behavior is a critical problem in several domains. Several theories have been developed by psychologists and social scientists over the last few decades to explain human behavior at the scale of a population. Notable among these theories are *The Theory of Planned Behavior* (Ajzen, 1991), *Social Cognitive Theory* (Bandura, 1986), *Self-Determination Theory* (Ryan & Deci, 2000) and the *Transtheoretical Model for Stages of Change* (Prochaska, 2008). There are several other theories that are specialized to different domains such as the *Health Behavior Model* (Cohen, Scribner, & Farley, 2000). While such models can explain aggregate behaviors at the scale of a population, these models are not actionable at the level of individuals.

Another efforts have resulted in fluid analogy models for human behavior that aim to operationalize the above theories in a control systems framework (Navarro-Barrientos, Rivera, & Collins, 2010; C. A. Martin et al., 2014; C. Martin, Deshpande, Hekler, & Rivera, 2015; Navarro-Barrientos, Rivera, & Collins, 2011; Dong et al., 2012). These models provide an intuitive and easy approach to separate the state variables and system parameters that drive the models. Such models have been effectively used in socially relevant programs for smoking cessation and health management (Lai, Cahill, Qin, & Tang, 2010). Despite their simplicity and effectiveness, there can be ambiguities in these models that limit their full exploitation in automated tools. For this reason, we are examining the utility of domain independent models that can be used to represent and reason about human behaviors.

Bond Graphs

Bond graphs were introduced in (Paynter, 1961) as a domain independent graphical representation to reason about systems involving mechanical, chemical and electrical components in a unified framework. In this approach, a system is viewed as comprising several components; each component has ports through which energy can be exchanged with other components. Every component is identified as being one that generates energy in the system or one that consumes energy. Components are connected through bonds between corresponding ports. Every bond has a half arrow that denotes which element in the bidirectional relationship generates energy and which element consumes energy. Energy transfer between components is viewed as a bidirectional exchange of *effort* and *flow* (Gawthrop, 1991; Broenink, 1999; Breedveld, 2008).

Client-Therapist Interaction Model

We view human behavior as one that involves complex energy transfers across multiple domains. In the context of our ongoing investigation into modeling human behavior for wellness management (Chippa, Whalen, Douglas, & Sastry, 2014; Mahamadi & Sastry, 2016b, 2016a), our interest is to develop actionable models for human behavior that can guide the decision-support. In this section, we present a model for the interaction between a Client and a Therapist that is inspired by the work in (Liebovitch, Peluso, Norman, Su, & J.M., 2011).

Figure 1 illustrates a fluid analogy model that represents the interaction between a client and a therapist. In this model, there are two tanks — one representing the client (right) and the other representing the therapist (left). Following (Liebovitch et al., 2011), the level of fluid in each of the tanks represent the *valence*, or *affect*, of the client (I_2) and the therapist (I_1), respectively. The valence of the therapist is a function of his or her training and is represented by the valve N_1. We assume that a better trained therapist, i.e., more flow in N_1, would have higher valence. The valence of the client is affected by the environmental conditions as represented by N_2. Through the interaction, the valence of the client and the therapist is changed because of the flows through the valves that are labeled Therapy (R_1) and Feedback (R_2).

We follow the procedure in our earlier work (Mahamadi & Sastry, 2016b) to construct the bond graph model, that is shown in Figure 2.

We derive the dynamic equations of the system :

$$\frac{d}{dt}(I_1) = N_1 \times S_1 - R_1 \times I_1 + R_2 \times I_2, \qquad (1)$$

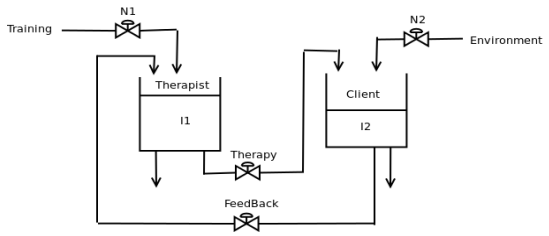


Figure 1: Fluid Analogy Model for the interaction between a Therapist and a Client.

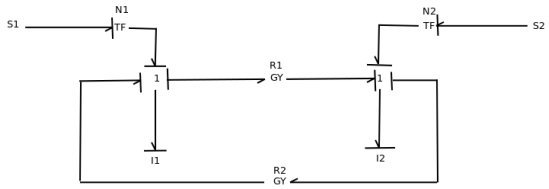


Figure 2: Bond Graph representation for the Therapist-Client model

and

$$\frac{d}{dt}(I_2) = N_2 \times S_2 - R_2 \times I_2 + R_1 \times I_1. \quad (2)$$

Maintaining Client Valence

The model in this paper is developed form the model in our previous work (Mahamadi & Sastry, 2016a) by adding a feedback from the client to the therapist. Now to maintain the Client valence, a well trained therapist should be able to control the flow of therapy to the client and the flow of feedback from the client. To model this interaction, we designed a controller to regulate the valves R_1 and R_2.

There are many options of controllers to be chosen for this control problem. However, we chose the proportional controller for the simplicity and the advantage of faster tuning.

After integrating the controller to the system, the transfer function of the system is demonstrated in Equation 3.

$$H(s) = \begin{bmatrix} \frac{20.02s}{25s^2+10s+20} \\ \frac{0.1s+20.02}{25s^2+10s+20} \end{bmatrix} \quad (3)$$

In order to test the design of the client valence regulator described above, we chose reasonable values for the training and the environment variables, then the response of the system to a unit set point is depicted in Figure 3.

We can conclude form Figure 3 a well trained therapist, has the ability to regulate the valence of the client by manipulating the rates of both the therapy and the feedback.

Examining the Client-Therapist Relationship

The model shown in Equation 1 and Equation 2 was also used to analyze the stability of the system. For example, we identified the critical points and plotted the system trajectories

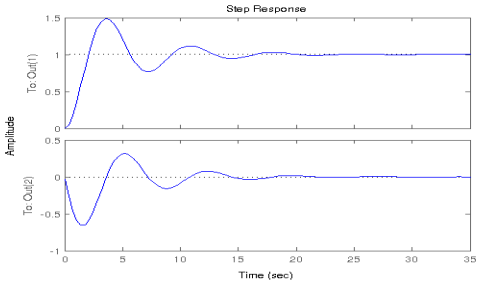


Figure 3: The step response of the client valence under the proportional controller

starting from different initial conditions, i.e., different initial states of both the client and the therapist. For example, if the therapist has initially positive valence, how is that going to affect the initially negative, neutral or positive client. Using the parameters from (Liebovitch et al., 2011) we obtained the phase portraits shown in Figure 4.

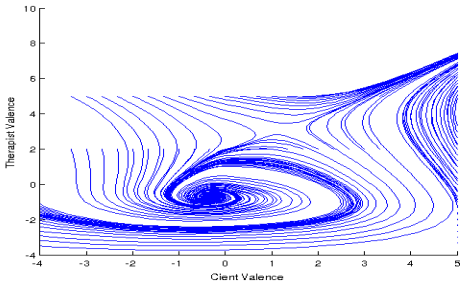


Figure 4: The phase portraits for the psychotherapy relation

Notice from Figure 4 that there are two critical points — the first is the stable point, *Attractor*, which is the point reached as a conclusion of a successful therapy program. The other point is the saddle point that represents a failed therapy. The figure shows that, when the therapist has a positive valence, the therapy sessions can lead a client with a negative or positive valence to the attractor. On the other hand a when the therapist has a negative valence, the session will conclude in the saddle point. These two stable points are similar to the ones reported in (Liebovitch et al., 2011).

Conclusions

The bond graph approach presented here to model human behavior is encouraging. Starting from a fluid analogy model for the interaction between a client and a therapist, we demonstrated that the bond graph approach yields a dynamic systems model that is similar to the one reported in the literature. As with any other model, one can analyze the behavior of the system and design controllers to achieve specific objectives.

References

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes.*, *50*, 179-211.

Bandura, A. (1986). *Social foundations of thought and action: A social cognitive theory* (N. Englewood Cliffs, Ed.). Prentice-Hall series in social learning theory.

Breedveld, P. C. (2008). Modeling and simulation of dynamic systems using bond graphs. In *Control systems, robotics and automation - modeling and system identification i.* EOLSS Publishers Co. Ltd./UNESCO.

Broenink, J. F. (1999). Introduction to physical systems modeling with bond graphs. In *in the sie whitebook on simulation methodologies.*

Chippa, M. k., Whalen, S. M., Douglas, F. L., & Sastry, S. (2014). Goal-seeking formulation for empowering personalized wellness management. In *Medical cyber physical systems workshop.*

Cohen, D. A., Scribner, R. A., & Farley, T. A. (2000). A structural model of health behavior: a pragmatic approach to explain and influence health behaviors at the population level. *Preventive medicine*, *30*, 146-154.

Dong, Y., Rivera, D., Thomas, D. M., Navarro-Barrientos, J. E., Downs, D. S., Savage, J. S., et al. (2012). A dynamical systems model for improving gestational weight gain behavioral interventions. In *American control conference.*

Gawthrop, P. (1991). Bond graphs: A representation for mechatronic systems. *Mechatronics*, *1*, 127-156.

Lai, D., Cahill, K., Qin, Y., & Tang, J. L. (2010). Motivational interviewing for smoking cessation. *Cochrane Database of Systematic Reviews*, *1*, CD006936.

Liebovitch, L., Peluso, P., Norman, M., Su, J., & J.M., G. (2011). Mathematical model of the dynamics of psychotherapy. *Cognitive Neurodynamics.*, *3*, 265-275.

Mahamadi, A., & Sastry, S. (2016a). A bond graph approach for modeling the client-therapist relation. In *The 2nd international conference on health informatics and medical systems.*

Mahamadi, A., & Sastry, S. (2016b). Bond graphs models for human behavior. In *Ieee international conference for basic sciences and engineering.*

Martin, C., Deshpande, S., Hekler, E., & Rivera, D. E. (2015). A system identification approach for improving behavioral interventions based on social cognitive theory. In *American control conference.*

Martin, C. A., Rivera, D. E., Riley, W. T., Hekler, E. B., Buman, M. P., Adams, M. A., et al. (2014). A dynamical systems model of social cognitive theory. In *American control conference.*

Navarro-Barrientos, J. E., Rivera, D. E., & Collins, L. M. (2010). A dynamical systems model for understanding behavioral interventions for weight loss. In *Sbp.*

Navarro-Barrientos, J. E., Rivera, D. E., & Collins, L. M. (2011). A dynamical model for describing behavioural interventions for weight loss and body composition change. *Mathematical and Computer Modelling of Dynamical Systems*, *17*, 183-203.

Paynter, H. M. (1961). *Analysis and design of engineering systems*. M.I.T. Press, Cambridge.

Prochaska, J. O. (2008). Decision making in the transtheoretical model of behavior change. *Medical Decision Making: an International Journal of the Society for Medical Decision Making*, *28*, 845-849.

Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, *55*, 68-78.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Modeling cognitive parsimony with a demand selection task

Othalia Larue (othalia.larue@wright.edu)
Department of Psychology, Wright State University
Dayton, OH 45435 USA

Ion Juvina (ion.juvina@wright.edu)
Department of Psychology, Wright State University
Dayton, OH 45435 USA

Keywords: demand selection; cognitive parsimony; cognitive modeling.

Introduction

The law of less work (Hull, 1943) is our natural tendency given two alternatives with equal incentives to pick the less demanding one. This notion also appears in the field of judgment and decision making (Gigerenzer & Goldstein, 1996; Tversky & Kahneman, 1974), it is referred to as internal cost of effort. Cognitive parsimony is our tendency to favour low-effort strategies that help us to decide faster and simple strategies to approach a complex problem. An experimental paradigm for this phenomenon has been developed by Kool, McGuire, Rosen, & Botvinick (2010) and referred to as the demand selection task. In this poster, we present a model of this task developed in the ACT-R architecture (Anderson, 2007), which offers an hypothesis as to which cognitive mechanisms might participate in this phenomenon.

Demand Selection Task

In the demand selection task (Kool et al., 2010), two decks of cards are placed symmetrically left and right of the center of the screen. The keyboard is used to select one of the decks and uncover the card upon which a digit, between 1 and 9, will be displayed**.** According to the color of the number, the subject has to perform a different type of judgement. Blue calls for a magnitude judgment: if the number is less than five, subjects should say yes, otherwise no. Yellow calls for a parity judgment: if the number is even, subjects should say yes, otherwise, no. Unbeknownst to the participants, one deck leads to a low demand task and the other deck to a high demand task. Participants are instructed to 'Feel free to move from one deck to the other whenever you choose' and 'if one deck begins to seem preferable, feel free to chose that deck the more often'. In the low demand task, the color of each numeral matches the previous color on 90% of trials, whereas in the high demand task, the color of each numeral matches the previous color on 10% of trials. Overall, response times (RT) and error rates showed that task switching was cognitively costly, and that subjects mostly choose to pick the less cognitively demanding deck. While some subjects demonstrated their awareness of this effect, the effect did not depend on their awareness of it, thus making the DST an interesting task to evaluate implicit behaviour.

Experimental procedure

We reproduced Experiment 1 from Kool et al's paper (2010). The simulation included 50 runs of 500 trials of the Demand Selection Task (DST). The task was self-paced with a maximum limit of time of 1h (which was never reached by the model or the participants). Subjects had to pick between two decks, by pressing a key ("F" for left, "J" for right). According to the color of the number (yellow or blue), participants had to either produce a parity judgment (even or odd) or a magnitude judgment (less or greater than five) on the number. Depending on the deck selected, the color of the number switched with a probability of 0.9 (making it a higher demand task) or 0.1 (making it a lower demand task) at each trial.

Model

The model[1] was built in the computational cognitive architecture and theory of human cognition ACT-R (Adaptive Control of Thought - Rational) (Anderson, 1990; 2007). In ACT-R, different modules, including two memory modules (procedural and declarative) interact to complete a cognitive task. The modules are accessed via their associated buffers. ACT-R has been used to model several tasks. Declarative memory stores facts about the environment (know what). The procedural memory, through procedural rules (know how), allows for action selection. ACT-R is a hybrid cognitive architecture composed of symbolic and subsymbolic components: the retrieval of a fact (symbol) from declarative memory depends on subsymbolic retrieval equations (pondering the context and history of retrieval of the fact), and, the selection of a rule (symbol) depends on utility subsymbolic equations (which computes costs and benefits associated to the rule). The memory elements (chunks) are reinforced through patterns of occurrence within the environment. Learning processes act at both subsymbolic and symbolic levels.

The preference of a deck over another one relies on implicit mechanisms: mainly base-level and spreading activation with the participation of utility learning. Base-level learning

[1] Model code available at: http://psych-scholar.wright.edu/astecca/software

determines how patterns of use affect chunk activation and decay. Spreading activation provides context to the retrieval since chunks will spread an amount of activation to other chunks in declarative memory, based on the relationship they have with other chunks. The choice between the two decks is represented by two procedures. After the model picks one deck, it perceives a number and a color, and then it retrieves the chunks associated to the color and the number. Chunks of the yellow color are associated with a 'parity' chunk, chunks of the 'blue' color are associated with a 'magnitude' chunk. The retrieved chunk is placed in the imaginal module. A judgement is produced based on the retrieved chunk, and an answer is vocalized. The chunk placed in the imaginal buffer will spread activation and influence the next retrieval request. A reward is back propagated after the answer has been produced. The failure to retrieve a judgment chunk will lead to errors which are also signaled to the model by backpropagation.

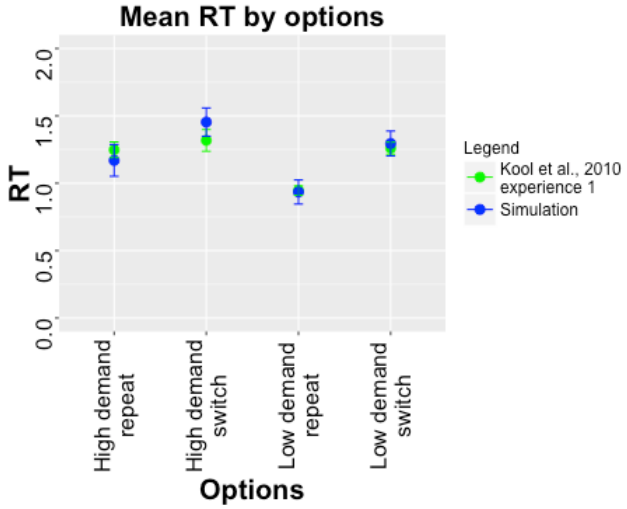


Figure 1: Mean RT by options.

Therefore, the gradual selection of the lower demanding deck occurs through two mechanisms: the retrieval of elements in the higher demanding deck (with high probability of switch) will take longer (representing the expended effort required) as activation from the previous trial will have spread less to the current trial. And, the longer this process takes, the less reward gets back-propagated to the selection of this deck (as the reward gets discounted with time). Thus, gradually, the selection of the less demanding deck is the one that is going to be reinforced the most. Errors encountered will be due to the failure of retrieval of judgment chunks.

Results

As in the original experiment, we measured the verbal RT for the two decks (low demand vs. high demand) and trial types (task switch vs. task repetition). Figure 1 shows the means of medians for each trial types and deck types. Table 1 shows the parameters used in the ACT-R model. An ANOVA indicated as in the original experiment significant effects for trial types ($F(1,50) = 9.940$; $p < 0.002$) and deck types ($F(1,50) = 3.691$; $p < 0.05$). Average selection of the lower demanding task is 63% in our experiment (68% in the original experiment).

Table 1: Model parameters.

Parameters	Value
:rt	-1.0
:alpha	0.1
:lf	1.5
:mas	3.0
:imaginal-activation	0.41
:ans	0.1
:bll	0.21

Discussion and conclusion

The demand selection task is aimed at evaluating the tendency to avoid cognitively demanding tasks. Computational cognitive models have been made of "minimal control" (Taatgen, 2007) and "least effort" (Anderson, 1990), but this is to our knowledge the first model of the DST. We were able to reproduce the results of Experiment 1 of Kool et al.'s paper (2010) with a simple ACT-R model. The performance at the DST in our explanation relies mainly on implicit mechanisms (utility learning and base-level and spreading activation), in accordance with experimental results showing that the participants did not need to be aware of the type of task (low demanding or high demanding) for the effect to be observed. The DST is interesting to correlate subjects' individual differences with their performance at different cognitive tasks. Having a model of such a task will allow us in future work to model individual differences as captured in the DST model and as they transfer into other tasks and might affect performance there (e.g. we are currently using this task in an ongoing research studying the relationship between cognitive parsimony and vulnerability to exploitation in interpersonal transactions).

Acknowledgments

This work was supported by The Air Force Office of Scientific Research grant number FA9550-14-1-0206 to IJ.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Anderson, John Robert. (1990). The adaptive character of thought: Psychology Press.

Gigerenzer, Gerd, & Goldstein, Daniel G. (1996). Reasoning the fast and frugal way: models of bounded rationality. Psychological review, 103(4), 650.

Hull, Clark. (1943). Principles of behavior. New York: Appleton-Century.

Kool, W., McGuire, J. T., Rosen, Z. B., & Botvinick, M. M. (2010). Decision making and the avoidance of cognitive demand. Journal of Experimental Psychology: General, 139(4), 665-682

Tversky, Amos, & Kahneman, Daniel. (1974). Judgment under uncertainty: Heuristics and biases. science, 185(4157), 1124-1131.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

A Computational Holographic Model of Memory for Abstract Associations

Matthew A. Kelly (Matthew.Kelly2@Carleton.ca)
Robert L. West (Robert.West@Carleton.ca)
Institute of Cognitive Science, Carleton University
1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6 Canada

Abstract

How do humans learn the syntax and semantics of words from language experience? How does the mind discover abstract relationships between concepts? Computational models of distributional semantics can analyze a corpus to derive representations of word meanings in terms of each word's relationship to all other words in the corpus. While these models are sensitive to topic (e.g., tiger and stripes) and synonymy (e.g., soar and fly), the models have limited sensitivity to part of speech (e.g., book and shirt are both nouns). By augmenting a holographic model of semantic memory with additional layers of representations, we demonstrate that sensitivity to syntax relies on exploiting higher-order associations between words. Our hierarchical holographic memory model bridges the gap between models of distributional semantics and unsupervised part-of-speech induction algorithms, providing evidence that semantics and syntax exist on a continuum and emerge from a unitary cognitive system.

Keywords: semantic memory; mental lexicon; latent semantic analysis; statistical learning; holographic models; memory; cognitive models; semantic space.

Orders of Association

Saussure (1916) defines two types of relationships between words: *paradigmatic* and *syntagmatic*. A syntagmatic relationship is the syntactic relationship a word has with other words that surround it. A paradigmatic relationship is when a pair of words can be substituted for each other. Building on Saussure, we define the term *order of association* as a measure of the degree of separation of two words in an agent's language experience.

First order association is when two words appear together. In the sentence "eagles soar over trees", the words *eagles* and *trees* have first order association. Words with strong first order association (i.e., frequently appear in the same sentence) are often related in topic.

Second order association is when two words appear with the same words. In the sentences "airplanes soar through skies" and "airplanes fly through skies", *soar* and *fly* have second order association. Words with strong second order association are often synonyms, or have a paradigmatic relationship in Saussure's terms.

Third order association is when two words appear with words that appear with the same words. Given the sentences in Table 1, the words *eagles* and *birds* have neither first nor second order association, but do have third order.

One can keep abstracting to higher-level orders of association indefinitely. At sufficiently higher orders of association, all words are related to all other words. Sensitivity to increasingly higher-order associations allows one to identify increasingly abstract relationships between items, such as syntactic categories. We hypothesize that to properly capture the syntagmatic relationships between words in the English language, it is necessary for a cognitive model to be sensitive to at least third-order associations.

Table 1: Artificial data set for Simulation 1.

Sentences
eagles soar over trees
birds fly above forest
airplanes **soar** through skies
airplanes **fly** through skies
airplanes **glide** through skies
dishes are **over** plates
dishes are **above** plates
dishes are **atop** plates
squirrels live in **trees**
squirrels live in **forest**
squirrels live in **woods**

Distributional Models of Semantics

Computational models of distributional semantics in the literature, such as LSA (Landauer & Dumais, 1997), HAL (Burgess & Lund, 1997), MINERVA 2 (Kwantes, 2005), the Topics Model (Griffiths, Steyvers, & Tenenbaum, 2007), and BEAGLE (Jones & Mewhort, 2007), are sensitive to only first and second-order associations. Jones and Mewhort observe clusters of vectors in semantic space that seem to correspond, roughly, to part of speech of information. Such clusters are suggestive of higher order associations, though BEAGLE does not exploit these higher order associations.

We have developed a model capable of detecting associations of arbitrarily high order. Using BEAGLE (Jones & Mewhort, 2007; but see also Kelly, Kwok, & West, 2015; Rutledge-Taylor et al., 2014, for variants on the model), a holographic model (Plate, 1995) of semantic memory, as a basis, we present a hierarchical model that layers multiple BEAGLE models. The *memory vector* outputs of one layer serve as the *environment vector* inputs to the next layer. The model roughly resembles a deep neural network in structure, but unlike a neural network, the model is not trained and the data is not subject to dimensional reduction at higher layers.

Simulation 1

Higher layers of the model are sensitive to higher orders of association (Figures 1 and 2), as demonstrated by an artificial data set (Table 1). The memory vectors for words

with second order association, such as *soar* and *fly*, are close on Layer 1 (cosine = 0.44) and draw closer in higher layers (Layer 4, cosine = 0.71). Whereas *eagle* and *bird*, which have only third order association, are distant on Layer 1 (see Figure 1, cosine = -0.08) but are close on Layer 2 (cosine = 0.16) and draw closer in higher layers (see Figure 2, Layer 4, cosine = 0.47). Figures show cosine distances between 256 dimensional vectors, the distances compressed to 3 dimensions by multi-dimensional scaling.

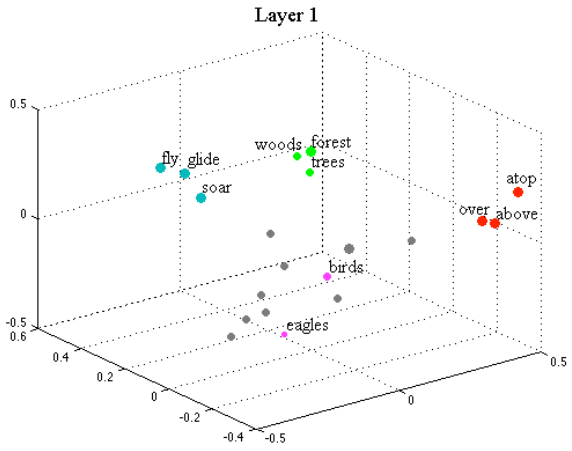


Figure 1: Cosine distances between vectors for Layer 1.

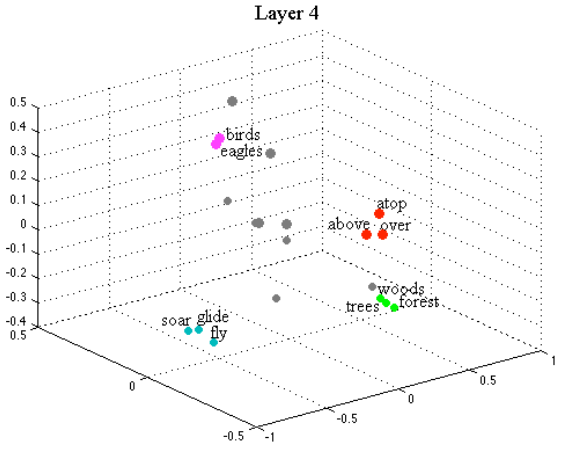


Figure 2: Cosine distances between vectors for Layer 4.

Simulation 2

We ran the model on 15 out of copyright books from the *The Bobbsey Twins* series of children's novels, available through Project Gutenberg. The corpus consists of 441 476 words and 9062 unique words. The model was run using 256 dimensional vectors and four layers. The model read the corpus one sentence at a time. Within each sentence, the model used a moving window of 21 words, 10 words to the left and right of a target word. Within that window, all *n-grams* are encoded as convolutions of environment vectors and summed into the target word's memory vector.

We find that higher layers of the model exploit higher-order associations to strengthen semantically and syntactically correct relationships only weakly present in Layer 1 (see Figure 3) and to suppress erroneously strong relationships present in Layer 1 (see Figure 4).

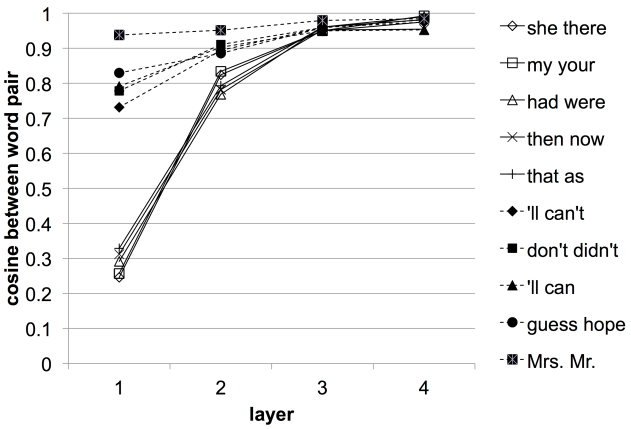


Figure 3: Cosine similarities between word pairs.

Word pairs are selected from the top 100 most similar word pairs on Layer 3 (Figure 3) or Layer 1 (Figure 4). Open marker and solid line indicates the 5 word pairs that increased in similarity the most from Layer 1 to Layer 3. Filled marker and dotted line indicates the 5 word pairs that increased in similarity the least from Layer 1 to Layer 3.

In Figure 3, we see a dramatic increase in similarity between word pairs such as the determiners *my* and *your*, from a cosine of 0.26 at Layer 1 to 0.99 at Layer 4. Conversely, word pairs such as the verbs *'ll* (contraction of *will*) and *can* or the titles *Mr.* and *Mrs.* are already highly similar at Layer 1 and so increase little across layers.

In Figure 4, we see that the similarity between dialogue tags *exclaimed* and *said* or the proper names of the twins *Bert* and *Nan* are strengthened by the higher order associations, whereas the erroneous relationship between the adjective *few* and *burgulor* (sic) or *thank* and the contraction *where'd* is suppressed.

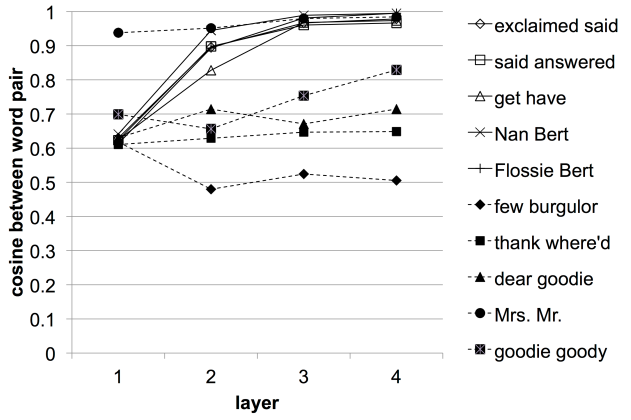


Figure 4: Cosine similarities between word pairs.

Acknowledgments

This research is supported by an Ontario Graduate Scholarship awarded to the first author and a grant from the Natural Sciences and Engineering Research Council of Canada to the second author.

References

Burgess, C., & Lund, K. (1997). Modelling parsing constraints with high-dimensional context space. *Language and Cognitive Processes, 12*, 177-210.

Jones, M. N., & Mewhort, D. J. K. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological Review, 114*, 1-37. doi: 10.1037/0033-295X.114.1.1

Griffiths, T. L., Steyvers, M., & Tenenbaum, J. B. (2007) Topics in semantic representation. *Psychological Review, 114*, 211-244. doi: 10.1037/0033-295X.114.2.211

Kelly, M. A., Kwok, K., West, R. L. (2015). Holographic declarative memory and the fan effect: A test case for a new memory model for ACT-R. In N. A. Taatgen, M. K. van Vugt, J. P. Borst, & K. Mehlhorn (Eds.), *Proceedings of the 13th International Conference on Cognitive Modeling (pp. 148-153)*. Groningen, the Netherlands: University of Groningen.

Kwantes, P. J. (2005). Using context to build semantics. Psychonomic Bulletin & Review, 12, 703-710.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review, 104*, 211–240.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks, 6*, 623–641. doi: 10.1109/72.377968

Rutledge-Taylor, M. F., Kelly M. A., West, R. L., & Pyke, A. A. (2014). Dynamically structured holographic memory. *Biologically Inspired Cognitive Architectures, 9*, 9-32. doi: 10.1016/j.bica.2014.06.001

Saussure, F. (1916). *Cours de linguistique générale*. C. Bally, A. Sechehaye, & A. Riedlinger, (Eds.). Lausanne, France: Payot. Feigenbaum, E. A. (1963). The simulation of verbal learning behavior. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought.* New York: McGraw-Hill.

In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

Concept Learning, Recall, and Blending with Regulated Activation Networks

Alexandre Miguel Pinto (ampinto@dei.uc.pt), **Rahul Sharma (rahul@dei.uc.pt)**
CISUC - Department of Informatics Engineering, University of Coimbra, Portugal

Herein we present the cognitive model Regulated Activation Networks (RANs), which aims at unifying the three perspectives (symbolic, connectionist, and geometric feature-space) of conceptual representations. It learns new concepts from input data, dynamically builds a hierarchy of abstract concepts, and learns the associations among them, both between different levels, and within the same level of the hierarchy. Its recall mechanism, the geometric backpropagation algorithm, allows the understanding of the meaning of higher level concepts in terms of input level features. The regulation mechanism we also introduce has a de-noising effect over the results obtained from the recall mechanism.

Keywords: Cognitive modeling, connectionism, dynamic systems, conceptual representations.

Design, Methodology and Approach

The Regulated Activation Networks (RANs) model is based upon the Principles laid out in (Pinto & Barroso, 2014), and its geometric interpretation is inspired from the theory of conceptual spaces(Gärdenfors, 2004) whereby concepts are regions in multidimensional spaces (dimension = feature). Topologically, a RAN is a connectionist model where each node represents one dimension/feature and its activation value represents the concept's value (in the interval $[0,1]$) along that dimension. An instance of the model is initialized with one layer of nodes – one node per input data feature – and dynamically builds new nodes and new layers solely driven by the complexity in the input data.

Inter-Layer Learning As the model's instance is exposed to input data, it resorts to some user-specified clustering algorithm to identify centroids of clusters in the data. The RANs model then creates one new node per centroid in a higher layer. The coordinates of each centroid are encoded as the inter-layer weights $w_{m,n}$ associated to the edges between the newly created centroid-node n and the nodes m in the lower input layer. After the creation of the second layer of nodes, each input datum (with values in the first layer of feature/node space) can be re-represented in the second layer of centroid/node space – we obtain this re-representation via our *upward activation propagation* algorithm.

Upward activation propagation This algorithm takes an activation pattern, i.e., the coordinate values, at layer L and calculates its normalized squared euclidean distance to each centroid in layer $L+1$. These distances are then passed through a non-linear radial basis function (in this paper we used $f(x) = (1-\sqrt[3]{x})^2$ but it can be replaced by any other similarly behaving function) that behaves as an activation/-transfer function – the smaller the distance, the higher the activation of the corresponding centroid. This results in an activation pattern in layer $L+1$ with one activation value for each of its centroid nodes. Figure 1 illustrates a RANs instance with two layers: L and $L+1$, where L has i nodes $(n_1, n_2, \ldots, n_i)$ with $(a_1, a_2, \ldots, a_i)$ corresponding activation values; and layer $L+1$ has j nodes $(N_1, N_2, \ldots, N_j)$ with $(a_1, a_2, \ldots, a_j)$ activations.

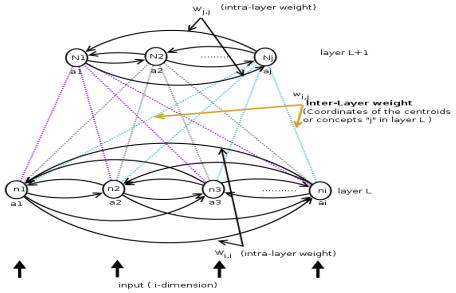


Figure 1: Learning in RANs

Intra-Layer Learning As lower layer L activation patterns get re-represented in the upper layer $L+1$ via upwards propagation, a pairwise correlation calculation takes place ate layer $L+1$: the intra-layer learning. These correlations are calculated via equation 1 and their values are stored as weights of the connections between the corresponding nodes.

$$W_{m\to n} = \frac{\sum_{k\in input\_set} [(1-|a_m^k - a_n^k|) - (1-a_m^k)*(1-a_n^k)]}{\sum_{k\in input-set} [1-(1-a_m^k)*(1-a_n^k)]} \quad (1)$$

In the numerator the part $(1-|a_m^k - a_n^k|)$ calculates the similarity of activations among nodes m and n, and the product $(1-a_m^k)*(1-a_n^k)$ is used to reduce the impact of the similarity when both activations are very close to 0 albeit similar.

Recall Concept recall amounts to obtaining, at the input feature space level, the representation of the selected higher concept node(s). Our geometric downward propagation algorithm works as follows: the user selects how strongly (s)he wishes to recall which higher layer concept(s) by injecting the corresponding activations Aj in their layer $L+1$; the algorithm generates a random activation pattern in layer L below, propagates it upward to obtain actual activation A'_j, and calculates the error $e_j = A'_j - A_j$; we use these individual errors to adjust the activation a_i of each node i in layer L below via $\Delta_{a_i} = (\sum_1^j \Delta_{a_i,A_j})/(\#j)$ where $\Delta_{a_i,A_j} = (W_{j,i} - a_i)*(e_j)$, with $W_{j,i}$ being the coordinate of centroid j in layer $L+1$ along dimension-node i in layer L. The overall impact of a_i on all A_j is summed together and normalized by dividing with

maximum possible impact i.e. $\#j$. Finally, the geometric error correction at node i of layer l is obtained by : if $\Delta_{a_i} >= 0$ then $a_i = a_i + \Delta_{a_i} * (1 - a_i)$; otherwise $a_i = a_i + \Delta_{a_i} * (a_i)$. The cycle <upwards propagation; error calculation; lower layer activation pattern correction via the error> is repeated until convergence of the lower layer activation pattern.

Regulation The recall results obtained are reasonable, but noisy. To denoise recall results a complementary Intra-Layer (IL) activation formula is developed which uses intra-layer weights to estimate each node's expected activation according to its same-layer companion nodes via

$$IL(a_n) = \frac{\sum_m \sigma_{m\to n}[(a_m * W_{m\to n}) + (1 - a_m) * (1 - W_{m\to n})]}{\sum_m \sigma_{m\to n}} \quad (2)$$

Here $W_{m\to n}$ is the intra-layer weight learned as in equation (1), and $\sigma_{m\to n} (= [2 * |W_{m\to n} - 0.5|]^2)$ is the impact factor of each correlation: $W_{m\to n=0.5}$ indicates high probability that node m has minimal (or no) impact over node n.

Our regulation mechanism uses IL activation producing a regulated activation $pr(a_n) = (1-\rho) * a_n + \rho * IL(a_n)$, where ρ is regulation-factor(a constant in $[0,1]$).

Experiments and Observations

First experiment We generated an artificial data set of 300 2-dimensional data points, c.f. Fig. 2.

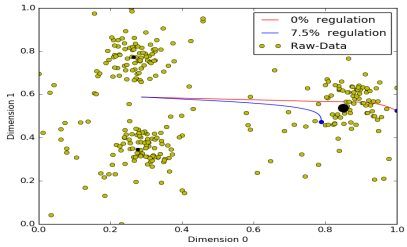


Figure 2: Observations on 2-D artificial data set

Setup The artificial data was generated such that it had 3 distinct clusters. We used K-means (MacQueen, 1967) to identify the clusters. The RANs model created 3 nodes in the first new layer (2nd layer), and 1 node in the second new layer (3rd layer), c.f., Fig. 3. To simulate recall we input the

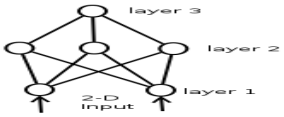


Figure 3: 3 layered model for artificial data

activation pattern $[1,0,0]$ in the second layer as expected activation *(in Fig. 2 black circles represents centers of clusters and their sizes depict expected activations)* and initiate the downwards propagation experiment.

Table 1: Observation of Artificial Data

Starting Layer 1 Act.	***Expected Layer 2 Act.***	**Regulation Factor (%)**	***Obtained Layer 2 Act.***
[0.28 0.58]	[1 0 0]	0	[**0.60** 0.1 0.12]
		7.5	[**0.68** 0.21 0.25]

Observations The algorithm randomly chooses a starting point ($[0.28, 0.58]$) and then repeats the ¡upwards activation propagation; error geometric downpropagation¿ cycle up to a maximum of 1000 iterations times; we did this for both with and without regulation. Fig. 2 shows the trajectories (each trajectory is a succession of points in the 2-D input feature space corresponding to the activations of the 2 bottom layer nodes) in 2-D. As per the expectation the trajectory obtained from regulation converges closer to the highly active center. Table 1 shows the activation at nodes in layer 2 corresponding to the converged points (without_regulation $[1, 0.52]$, with_regulation $[0.78, 0.47]$) in layer 1.

Blending Experiment with the MNIST data set We performed the experiment for blending (simultaneous recall of multiple concepts resulting in their fusion) using the MNIST (*The MNIST database of handwritten digits*, n.d.) data set with 250 images, and K-means which identified 31 clusters whose centroids are shown in Fig. 4. We create just two layers to show the concept blending operation – layer 1 has 784 nodes representing the pixels, layer 2 has 31 nodes.

Figure 4: Images represented by nodes in layer 2

Blends are obtained by injecting full activation (1) at nodes in layer 2 representing the concepts to blend, and by downwards geometric backpropagation. E.g., injecting 1 at nodes 6, 8, 25, 27, and 28 (these nodes correspond to images of digits 2 and 5, c.f., Fig. 4), and zeros in others we obtain the blend shown in the left-most image of Fig.5. In Fig. 5, the 2nd left image is a blend of 4's and 9's, the 3rd is a blend of 8's and 3's, and the last is a blend of 5's and 3's. These are not mere superpositions of the original clusters.

Figure 5: Blend of different centers

Acknowledgment

This work was supported by the project ConCreTe. The project ConCreTe acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 611733.

References

Gärdenfors, P. (2004). *Conceptual spaces: The geometry of thought*. MIT press.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth berkeley symposium on mathematical statistics and probability, volume 1: Statistics* (pp. 281–297). Berkeley, Calif.: University of California Press. Retrieved from http://projecteuclid.org/euclid.bsmsp/1200512992

The mnist database of handwritten digits. (n.d.). Retrieved from http://yann.lecun.com/exdb/mnist/

Pinto, A. M., & Barroso, L. (2014, July). Principles of regulated activation networks. In N. Hernandez, R. Jäschke, & M. Croitoru (Eds.), *Graph-based representation and reasoning: ICCS 2014* (Vol. 8577, pp. 231–244). Iaşi, Romania: Springer. doi: 10.1007/978-3-319-08389-6_19