Title
jTRACE: A Reimplementation and Extension of the TRACE Model of Speech Perception and Spoken Word Recognition

Journal
Proceedings of the Annual Meeting of the Cognitive Science Society, 27(27)

ISSN
1069-7977

Authors
Harris, Harlan D.
Magnuson, James S.
Strauss, Ted

Publication Date
2005

Peer reviewed

jTRACE: A Reimplementation and Extension of the TRACE Model of Speech Perception and Spoken Word Recognition

Ted J. Strauss (ted.strauss@uconn.edu)
James S. Magnuson* (james.magnuson@uconn.edu)
Harlan D. Harris (harlan.harris@uconn.edu)
Department of Psychology, University of Connecticut, 406 Babbidge Road, Unit 1020
Storrs, CT 06269 USA

Abstract

This paper describes jTRACE, a freely-available, cross-platform reimplementation of the TRACE model of speech perception and spoken word recognition in Java. The goal of the reimplementation is to facilitate the use of simulations by researchers who may not have the skills necessary to use the original C implementation of TRACE. In this paper, we report a large scale validation project, in which we have replicated a number of important previous simulations, and then we describe several new features in jTRACE designed to help researchers conduct original TRACE research as well as to replicate earlier findings. These include visualization tools, powerful scripting, built-in data analysis and graphing, stochasticity, and save/load functions that facilitate archiving and sharing simulations.

Overview

TRACE (McClelland and Elman, 1986) is arguably the best psychological model of speech processing to date, as it is able to simulate the deepest and broadest range of empirical phenomena. However, it is not widely used. One obstacle is that the original implementation in the C programming language is opaque to the average psychologist (or even the average programmer). We present jTRACE, a user-friendly, cross-platform and free software tool that reimplements the TRACE model in the Java programming language. Researchers at different technical levels have different modeling needs. jTRACE accommodates most of these needs, hiding details from the beginner, giving powerful scripting tools to the advanced user, and providing a basis for easy extensibility by programmers.

The introduction gives a primer on TRACE and the motivations for creating this tool. The second section reviews some old and new simulations that we have replicated with jTRACE. The third section describes the principal functions that make this an effective and versatile tool. Readers are encouraged to download jTRACE from:

`http://maglab.psy.uconn.edu/jtrace.html`

Introduction

In their seminal paper, McClelland and Elman (1986) describe TRACE and successful modeling of human behavior on a wide array of speech processing tasks. Prior to TRACE, the Cohort model (Marslen-Wilson & Tyler, 1980) laid the groundwork for modeling spoken-word recognition in real-time. TRACE takes the main ideas of Cohort theory and implements them within the Parallel Distributed Processing framework (McClelland & Rumelhart, 1981), thus allowing the model to be implemented as a computer program, and generate detailed, falsifiable predictions about human behavior. TRACE successfully models many aspects of real-time spoken language processing, including categorical perception of phonemes, word segmentation, lexical effects on phoneme processing, lexical competition (McClelland & Elman, 1986), and even fine-grained time course data from eye tracking (e.g., Allopenna, Magnuson, & Tanenhaus, 1998; Dahan, Magnuson, & Tanenhaus, 2001; Dahan, Magnuson, Tanenhaus, & Hogan, 2001), among many other phenomena (see Protopapas, 1999, for a review of TRACE and its place in the literature, and Frauenfelder & Peeters, 1998, for analyses of various TRACE parameters).

How TRACE Works The TRACE model is a connectionist network with an input layer and three processing layers: feature, phoneme and word. There are bottom-up excitatory connections between input-feature, feature-phoneme and phoneme-word layers. There are lateral inhibitory connections between units within the feature, phoneme and word layers. There are top-down excitatory (feedback) connections between word-phoneme and phoneme-feature layers (although phoneme-feature feedback is typically set to 0.0; McClelland & Elman, 1986). An external stimulus is given to the input layer and on each processing cycle, and activation passes along the weighted connections, changing the activation values of units in the processing layers.

The input to TRACE is a pseudo-spectral representation (McClelland & Elman, 1986). The input takes the form of a 63-dimensional vector at each time increment. Each time increment in TRACE is intended to approximate about 10 milliseconds of real time, and the 63-dimensional vector describes the activation of 7 acoustic features, each comprised of 9 continua. Feature units have excitatory connections to phonemes. TRACE uses a 14 phoneme subset of English phonemes plus a "silence" phoneme: /p/, /b/, /t/, /d/, /k/, /g/, /s/, /S/, /r/, /l/, /a/, /i/, /u/, /^/, and the silence phoneme, /-/. Phoneme units have excitatory connections to word units. Simulations can be run with no items in the lexicon, just a few, or hundreds, depending on simulation needs (simulations are faster with smaller lexicons). The original, standard lexicon (slex) has 212 items, though fairly large lexicons have

* Also at Haskins Labs, New Haven, CT.

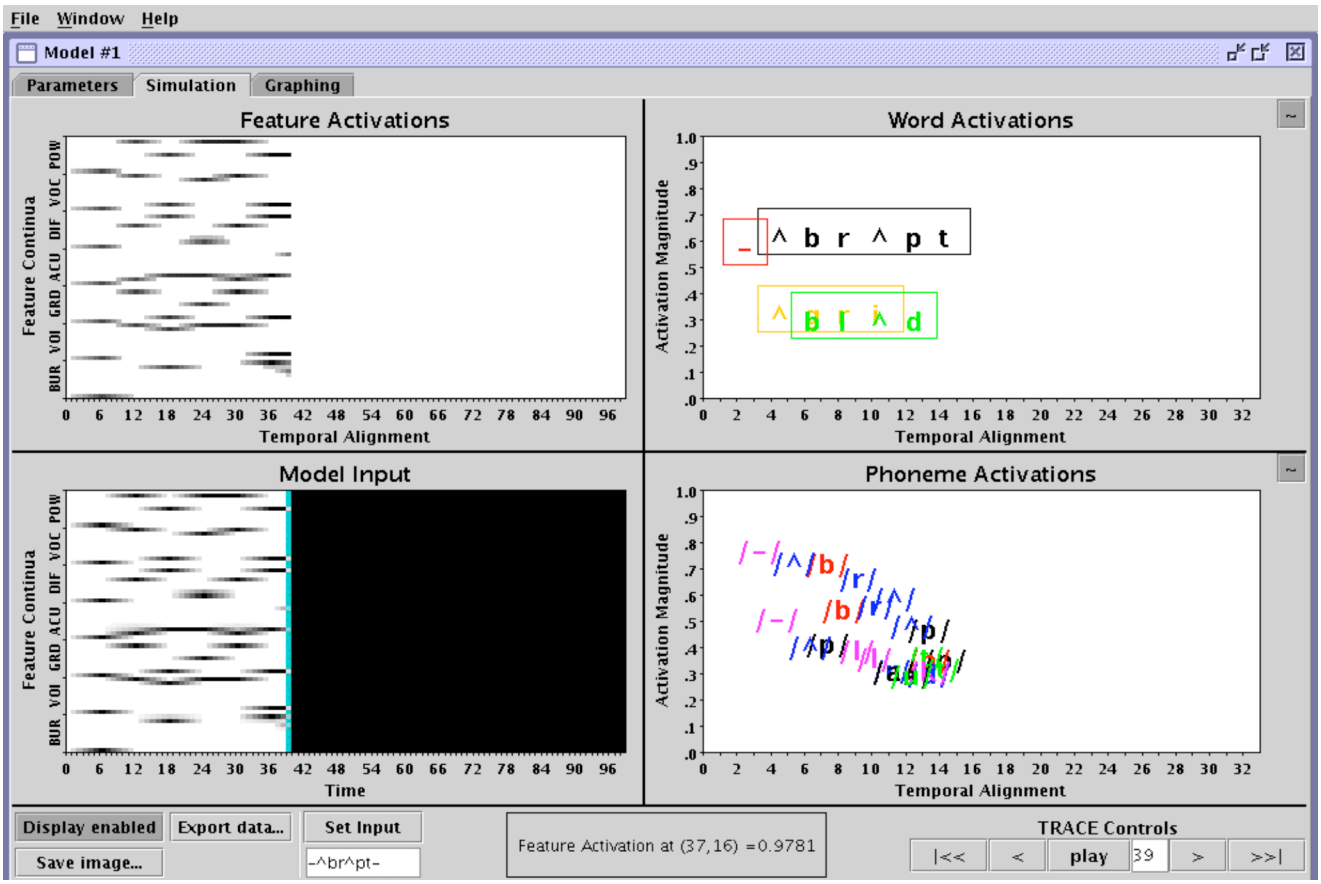

Figure 1: The jTRACE simulation panel, during presentation of "abrupt" (at about the middle of the /t/). Word and phoneme symbols are color coded in the two right panels in jTRACE. The right two panels can also be toggled to show grey-scale matrix representations like those in the two left panels. The solid black area in the lower left panel indicates time slices where no input has yet occurred.

also been used (Frauenfelder & Peeters, 1998, used 1024 words, and Magnuson, Strauss, & Harris, 2005 [this volume] used 901).

The user need only provide a string of phonemes and silence units and the model converts these into blocks of pseudo-spectral activation values. To illustrate what these inputs are like, consider Figure 1, which is a screen shot of the jTRACE simulation panel. The bottom-left window shows the input activations, where the x-axis represents time, the y-axis represents the 63 acoustic feature values, and the darkness of the squares corresponds to the magnitude of activation. The stimulus in this example is the phoneme sequence /-^br^pt-/ which corresponds to the English word "abrupt" with a short silence at both ends.

TRACE has a very particular way of representing time in the phoneme and word layers, which has been the target of substantial criticism (though the problems were first noted by McClelland & Elman, 1986). Each word and each phoneme is represented by a series of overlapping units aligned with different time increments. For example, when TRACE processes /-^br^pt-/ it will "hear" /^/ occurring once early on and again later in the word. The phoneme layer has multiple /^/ units distributed evenly across time that allow TRACE to activate the phoneme simultaneously at different temporal positions. The same temporal representation exists in the word layer, allowing TRACE to interpret its input as words distributed over time. In practical terms, this means the layers representing words and phonemes are duplicated every 3 time slices (so for a 100-cycle simulation, there are 33 copies of each phoneme and each word). This brute force approach is key to TRACE's solution to the segmentation problem. Rather than explicitly trying to detect word onsets, words are implicitly recognized. The word unit that is both most similar to the input and best aligned with it in time will become highly active and inhibit other units.

While TRACE's phoneme and word units are aligned with temporal positions, a TRACE simulation also occurs over a period of time, measured in processing cycles. So there are two notions of time associated with simulations: temporal alignment of phoneme and word units, and processing time – i.e., the temporal duration (or temporal extension) of each simulation, which proceeds in discrete *cycles*.

Every TRACE simulation begins at cycle zero, when no input has been presented to the model, and extends over time. At successive time steps, the input vectors are applied to the input units, which feed forward to the feature units, providing an analog to the temporal nature

of real speech. The input is processed in 8 consecutive steps as follows: (1) input to feature activation, (2) lateral feature inhibition, (3) feature to phoneme activation, (4) lateral phoneme inhibition, (5) phoneme to feature feedback activation (if turned on), (6) phoneme to word activation, (7) word to phoneme feedback activation, (8) lateral word inhibition. The feature, phoneme and word activations also decay linearly in time. These steps are calculated for each time cycle and progress is stored in each layers' activation values (see McClelland & Elman, 1986, for details about all parameters).

Behavioral claims about TRACE simulations are usually made with respect to the phoneme and word layers' activations over time. The right side of Figure 1 shows the phoneme and word layer activations, in which units surpassing an activation threshold are displayed. In the phoneme layer (bottom-right) the magnitude of activation for an individual phoneme unit is represented by its vertical height and its alignment in time is represented by its horizontal position. Therefore, two or more phoneme units aligned to the same time slice with comparable activations would reflect ambiguity as to the identity of the presented phoneme. The word layer is represented the same way: height equals activation and horizontal position equals temporal alignment. In the word window we see that TRACE has more-or-less identified the input as the lexical item "abrupt", although "agree" and "blood" are still somewhat activated.

TRACE, as an interactive activation network, is not a learning model. All connection weights and other processing parameters are set by hand before running a simulation. However, a user will typically adjust a few or no parameters from their default values when setting up a simulation.

We have just described how TRACE simulates speech perception and spoken word recognition. TRACE was originally implemented by McClelland and Elman as a C program (called cTRACE hereafter) which has been the basis of all TRACE research to date. cTRACE is run as a UNIX command line program; parameters and simulation data are saved to text files and analyzed separately from the program. cTRACE is efficient, can be used effectively by UNIX users with some programming expertise, and has contributed to many publications. The next section explains why we found it necessary to move on from the cTRACE implementation of TRACE.

Motivations for creating jTRACE. We had two primary motivations for creating jTRACE. First, the original code is fairly difficult to use and quite difficult to change. cTRACE was implemented in the early 1980s in C. While a technically proficient psychologist can use the main functions of the program, even minor extensions to the model require significant programming skill and considerable time. Furthermore, the cTRACE code is sparsely commented and does not conform to modern programming style, and this makes extensions to the model difficult to implement. The existence of a user-friendly TRACE program that includes visualization tools would encourage wider use of the model by non-specialists and researchers in related fields. Similarly, a *programmer*-friendly implementation would encourage researchers to extend TRACE in new ways. jTRACE meets both these needs.

Our second motivation was that many researchers hold strong expectations about what TRACE should predict for particular tasks. But because it is so difficult to conduct TRACE simulations, researchers often test these intuitive predictions with experiments with human subjects without confirming them via simulation. Quite often, these intuitive predictions turn out to be wrong – in the sense that the intuitions are wrong, not the model (see, for example, the discussion of predictions about frequency effects in Dahan et al., 2001). If an easy-to-use version of TRACE were available, we expect more researchers would be able to use actual simulations to generate TRACE predictions. However, the ability to run simulations is not a panacea; see Magnuson, Dahan and Tanenhaus (2001) and Mirman, McClelland, and Holt (2005) for cautionary examples of the importance of linking TRACE and human tasks as closely as possible, and the importance of evaluating what aspects of a simulation are responsible when the model fails to predict human data.

All the same, jTRACE improves the situation by allowing users to set up and run TRACE simulations in minutes without any knowledge of programming. Once set up, simulation parameters can be saved to a simple output files (in XML format) and shared with colleagues. In this way, reported simulations can be replicated and analyzed more easily, encouraging understanding of the model's behavior, as reported throughout the literature. jTRACE comes preloaded with several foundational simulations (see below), making it ideal for self or classroom tutorial use.

Validating jTRACE

This section summarizes a number of TRACE simulation replications that we have performed using jTRACE. Each of these simulations was originally done using cTRACE or an extension thereof. We will describe the validation process and measures, and then summarize the simulations.

We used two methods to validate jTRACE simulations against their cTRACE equivalents. The first and most comprehensive involved comparing the activation of every unit in each layer in the two simulations. In these cases identical simulation parameters were loaded into both cTRACE and jTRACE and the activation values of the two simulations were compared directly using a difference metric we call *scaled mean absolute difference* (SMAD). The difference metric results in a value between 0 and 100 that summarizes the difference between the two simulations; 0 means the two simulations are identical and 100 means they are maximally different (e.g., for a unit that can have activation ranging from –0.3 to 1.0, maximally different would mean the unit had a value of –0.3 in one implementation but 1.0 in the other). This direct comparison of cTRACE and jTRACE results was made possible by implementing the difference metric in

jTRACE as well as implementing a function that reads in data files created by cTRACE.

The SMAD procedure operates by reading in the two sets of simulation data, cSIM (from cTRACE) and jSIM (from jTRACE). The two sets are 4-dimensional arrays of real numbers having identical dimensions. The exact dimensions of a simulation depend on parameters, but a typical set of simulation dimensions (for a 100-cycle simulation) would be: feature layer (63 feature continua x 100 slices); phoneme layer (15 phonemes x 33 slices, since phonemes are aligned with the input every 3 slices); word layer (212 words x 33 slices). All of the latter are iterated over 100 processing cycles, yielding a total array size of 1,379,100 real numbers for each simulation. Processing this many comparisons requires approximately 5 minutes in jTRACE.

Each unit's activation is a real number ranging between –0.3 and 1.0, TRACE's standard minimum and maximum activation values. The SMAD metric compares each corresponding pair in cSIM and jSIM by subtraction, and the absolute value is taken for each. The mean of all absolute differences is computed. Since the mean absolute difference must be a number between 0 and 1.3, we scale it to range between 0 and 100 for clarity, and interpret this scaled mean absolute difference as a percent difference between the two simulations.

The SMAD metric was applied to all of the individual simulation replications and the percent difference was never greater than 0.007%. The largest individual unit difference (the absolute difference between a particular pair of cells in the two data sets cSIM and jSIM) was below 3%, although the majority of simulation comparisons had a maximum unit difference below 1%. We conclude that a difference between each simulation pair of less than 0.007% is small enough to conclude that jTRACE is a faithful reimplementation of cTRACE. The minor differences between the two stem from some necessary algorithmic changes (e.g., cTRACE relies heavily on pointer arithmetic, which is not available in Java) and subsequent rounding differences. (A more thorough treatment of the SMAD and difference data is presented in Strauss et al., in preparation.)

The SMAD validation method was applied for Replications 1-12 (see Table 1), which were drawn from the original TRACE paper and represent the core evidence used by McClelland and Elman to argue for the interactive activation framework. The parameter and stimulus sets for each of these simulations are bundled with the jTRACE download file [may not be hard-coded, but will be provided -- some of these will be scripts, and some will be single simulations]. Each of these replications involved between one and 213 TRACE simulations. The SMAD column for each gives the largest difference observed for each replication.

The second validation method applies to replications that involve doing multiple simulations and then averaging over their respective data, or doing another analysis that yields a multi-simulation result. In these cases (Replications 13-15), we compared the final aggregate data, and found virtually no differences between the two implementations in each case.

Table 1: Simulations with the original version of TRACE that have been replicated with jTRACE. SMAD is a measure of difference between jTRACE and the original TRACE implementation (cTRACE); see text for details.

	Simulation description	SMAD
1.	Basic lexical effect on phonemes (McClelland & Elman, 1986, p.24)	0.0016
2.	Elimination of the lexical effect by time pressure (*ibid*, p.26)	0.0009
3.	Late lexical effects (*ibid*, p.27)	0.0023
4.	Dependence of lexical effect on phonological ambiguity (*ibid*, p.28)	0.0011
5.	Lexical effects in reaction-time studies (*ibid*, p.29)	0.0036
6.	Absence of lexical effect in some reaction-time studies (*ibid*, p.30)	0.0031
7.	"Lexical conspiracy" (*ibid*, p.33)	0.0022
8.	Time-course of word recognition effects (*ibid*, p.57)	0.0022
9.	Lexical basis for word segmentation (*ibid*, p.62)	0.0017
10.	Word segmentation and non-words (*ibid*, p.65)	0.0011
11.	Recognizing words in short sentences (*ibid*, p.69)	0.0067
12.	Recognition of all items in the slex lexicon (*Ibid*, p.62)	0.0055
13.	Phoneme context effect with a stochastic version of cTRACE (McClelland. 1991, Figure 9)	n/a, stochastic data
14.	Time-course of frequency effects in eye-tracking experiment (Dahan et al., 2001, Fig. 6)	n/a, aggregate data
15.	Effects of lexical feedback in TRACE (Frauenfelder & Peeters, 1998, Figure 4.9; see also Magnuson et al., 2005)	n/a, aggregate data

Consider replication 13, which comes from McClelland's (1991) stochastic extension of TRACE, in which Gaussian noise could be added to both the input and to the interactions between processing units. This extension of TRACE simulated effects of lexical context on phoneme perception that the non-stochastic version could not. To replicate McClelland's result, stochasticity was implemented in jTRACE and can now be used with any simulation. To validate, we replicated McClelland's 4500 simulations and analyses. Because the output from TRACE is stochastic in this case, it does not make sense to compare individual simulations. Instead, we compared the aggregate overall results.

Replication 14 required adding three different implementations of frequency. Dahan et al. (2001) conducted eye-tracking experiments to assess the contribution of lexical frequency to word recognition throughout the time-course of lexical activation. After obtaining data supporting the claim that frequency affects lexical access from earliest moments of processing, they modeled their results in a version of TRACE that was

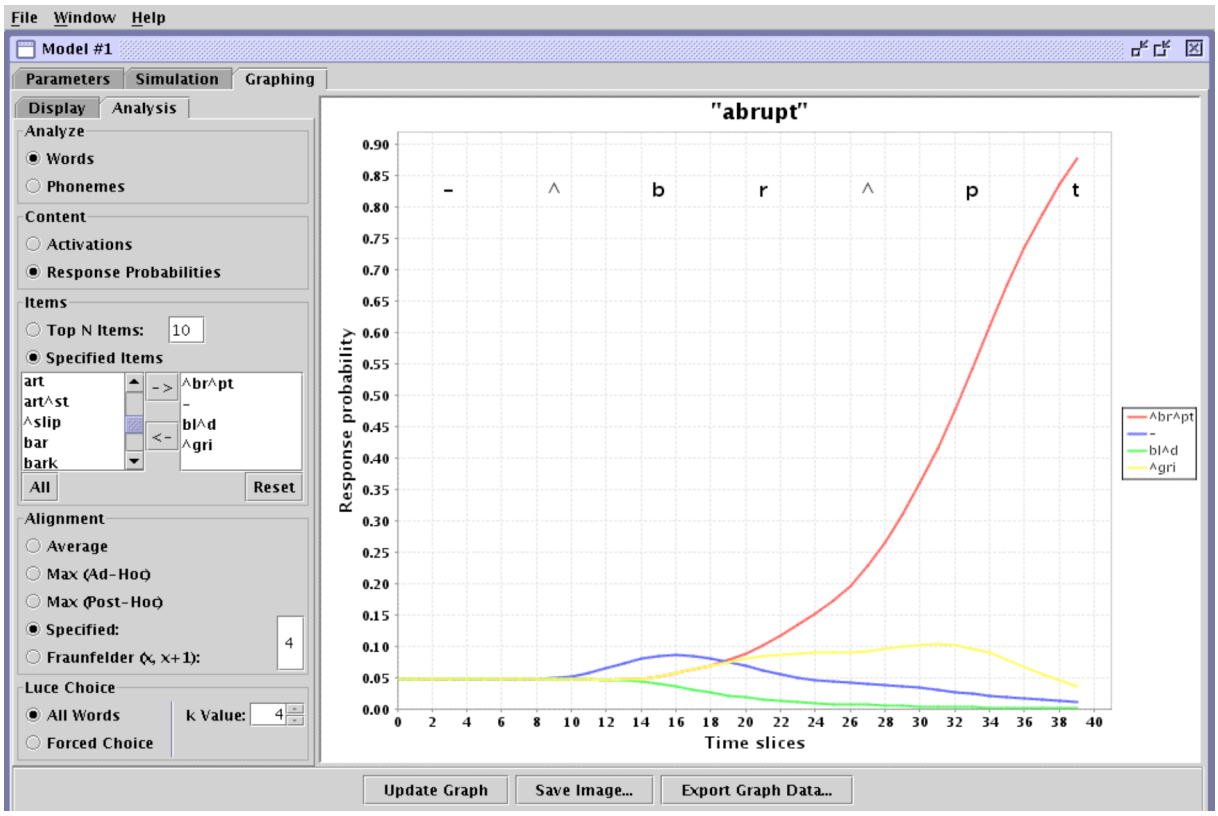

Figure 2: The graphing panel, showing Luce choice rule response probabilities as "abrupt" is presented to jTRACE. Note that the lines are color-coded in jTRACE.

extended to incorporate frequency effects. Each word in TRACE's lexicon is given a frequency value (based on a corpus). Three implementations were compared by Dahan et al. (resting levels, connection strengths, and post-lexical bias) jTRACE now includes the 3 frequency rules as normal parameters and we used them to replicate the simulations of Dahan et al. (2001).

The final replication listed in Table 1 is Frauenfelder and Peeters' (1998) investigation of the lexical feedback parameter. Frauenfelder and Peeters investigated whether feedback helps TRACE recognize lexical items more quickly. They found that feedback did not consistently help; about half the 21 items they tested were recognized more quickly *without* feedback (but see Magnuson et al., 2005 [this volume], who report that when the entire lexicon is examined, 73% of items are recognized more quickly with feedback, and that feedback makes the model more accurate and faster when noise is added to the input).

jTRACE's Features

This final section describes 10 of jTRACE's features that make it powerful, versatile and user-friendly. These features have greatly facilitated the replications described in Table 1, and currently are leading to the development of new TRACE modeling results in a few labs.

1. First and foremost is the addition of a graphical user interface, as shown in Figures 1 and 2. A jTRACE simulation "document" consists of the parameters of a simulation and the options used to visualize it. Each simulation's parameters can be easily modified, the simulation can be run with real-time visualization of activation levels, and the results can be interactively analyzed by creating graphs of activation and response probabilities of words and phonemes. Additionally, a scripting window allows complex sets of simulations to be run in a batch mode, with results exported to files for further analysis or displayed within jTRACE.

2. jTRACE has been coded with modern programming practices, and should be easy for programmers to understand and enhance. A programmer's guide is available from http://maglab.psy.uconn.edu/jtrace.html.

3. The jTRACE graphing panel (Figure 2), generates flexible graphs of phoneme or word unit activations or Luce choice rule response probabilities (Luce, 1959). Several options for how to calculate this measure are implemented and can be adjusted on the fly. A process that previously required custom programming or hours with a spreadsheet is now done automatically.

4. The scripting panel (not shown) automates simulation preparation, execution and analysis. Groups of simulations can be performed with iteration over ranges of parameter values, input values, phoneme continua or even analysis settings. A range of options is available for saving and formatting data. The scripting function is the most powerful feature in jTRACE, and easy to use.

5. Two categories of decision rules have been implemented in the scripting function based on work by Frauenfelder and Peeters (1998). A decision rule helps

provides a linking hypothesis between word and phoneme activations and human data.

6. In the simulation panel, word activations and phoneme activations can be visualized as "floating units", as pictured in Figure 1, wherein a unit's vertical height corresponds to the magnitude of its activation. This type of visualization is based on the diagrams of the original TRACE paper. In jTRACE, as the simulation runs, the floating units are animated, showing word and phoneme activations evolving in time. Matrix representations like those shown in the left panels of Figure 1 can also be used for words and phonemes.

7. As reported in the replications section, several extensions to the original TRACE model have been implemented in jTRACE. These extensions are input noise and unit stochasticity as per McClelland (1991), three implementations of frequency as per Dahan et al. (2001), and the decision rules and particular Luce choice rule calculations described in Frauenfelder and Peeters (1998).

8. Because jTRACE was implemented in Java, the program is completely platform-independent. (Most personal computers already have Java installed for web applications; Java is easily downloaded and installed.)

9. jTRACE can create several types of data file for analysis, sharing, and archival purposes. Raw results of simulations can be saved to text files for external analyses. Results of built-in analyses can be exported. Graphical representations of the simulation and analysis can also be saved for presentations and publications.

10. jTRACE has the ability to load simulation data generated by cTRACE (an augmented version of cTRACE that can export activation information in a format that jTRACE can read is available at the jTRACE web site). This allows comparison between the two models for any simulation based on the original parameters. Thus, any researchers with doubts about the accuracy of jTRACE as a reimplementation have the ability to make their own comparisons between the two programs.

11. jTRACE is freely available, and distributed under an open source license. We hope that users will tweak existing features and add new ones – not just to the interface and tools, but to the underlying model, and that TRACE will continue to inspire research in speech perception and spoken word recognition.

Conclusions

As TRACE continues to stimulate healthy debate in the field, the need for an easy-to-use, platform independent tool that can be used for education, replications, and full-scale modeling tasks is apparent.

Furthermore, as new experimental results challenge the modeling capabilities of the original TRACE model, increasingly complex extensions to the model are being proposed and implemented. jTRACE offers a framework wherein diverse types of extensions to the original model can be implemented and combined with one another via simple parameters.

Acknowledgments

Supported by National Institute on Deafness and Other Communication Disorders Grant DC-005765 to JSM.

References

Allopenna, P. D., Magnuson, J. S., & Tanenhaus, M. K. (1998). Tracking the time course of spoken word recognition using eye movements: Evidence for continuous mapping models. *Journal of Memory and Language, 38,* 419-439.

Dahan, D., Magnuson, J.S., & Tanenhaus, M.K. (2001). Time course of frequency effects in spoken-word recognition: evidence from eye movements. *Cognitive Psychology*, 42, 317–367.

Dahan, D., Magnuson, J.S., Tanenhaus, M.K., and Hogan, E.M. (2001). Tracking the time course of subcategorical mismatches: Evidence for lexical competition. *Language & Cognitive Processes, 16*, 507-534.

Frauenfelder, U. H. & Peeters, G. (1998). Simulating the time course of spoken word recognition: an analysis of lexical competition in TRACE. In J. Grainger and A. M. Jacobs (Eds.), *Localist connectionist approaches to human cognition* (pp. 101-146). Mahwah, NJ: Erlbaum.

Ganong, W. F. (1981). Phonetic categorization in auditory word perception. *Journal of Experimental Psychology: Human Perception & Performance*, 6, 110-125.

Luce, R. D. (1959). *Individual choice behavior*. New York: Wiley.

Magnuson, J. S., Dahan, D., & Tanenhaus, M. K. (2001). On the interpretation of computational models: The case of TRACE. In J. S. Magnuson & K. M. Crosswhite (Eds.), *University of Rochester Working Papers in the Language Sciences*, 2 (1) , 71–91.

Magnuson, J. S., Strauss, T. J., Harris, H. D. (2005) Interaction in spoken word recognition models: feedback helps. To appear in the *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Marslen-Wilson, W. & Tyler, L. K. (1980). The temporal structure of spoken language understanding. *Cognition*, 8, 1-71.

McClelland, J.L. (1991). Stochastic interactive processes and the effect of context on perception. *Cognitive Psychology*, *23*, 1-44.

McClelland, J.L. & Rumelhart, D.E. (1981). An interactive activation model of context effects in letter perception, Part 1: An account of basic findings. *Psychological Review*, *88*, 375-407.

McClelland, J.L., & Elman, J. L. (1986). The TRACE model of speech perception. *Cognitive Psych.*, *18*, 1-86.

Mirman, D., McClelland, J. L., & Holt, L. L. (2005). Computational and behavioral investigations of lexically induced delays in phoneme recognition. *Journal of Memory and Language, 52,* 416-435.

Protopapas, A. (1999). Connectionist modeling of speech perception. *Psychological Bulletin*, 125, 410-436.

Strauss, T.J., Magnuson, J. S., & Harris, H. D. (in preparation). jTRACE: A reimplementation and extension of TRACE for research and education.