Behavior Research Methods
2007, 39 (1), 19-30

jTRACE: A reimplementation and extension of the TRACE model of speech perception and spoken word recognition

TED J. STRAUSS
University of Connecticut, Storrs, Connecticut

HARLAN D. HARRIS
New York University, New York, New York

AND

JAMES S. MAGNUSON
University of Connecticut, Storrs, Connecticut
and Haskins Laboratories, New Haven, Connecticut

This article describes jTRACE, a freely available, cross-platform Java reimplementation of the TRACE model of spoken word recognition. The goal of the reimplementation is to facilitate the use of simulations by researchers who may not have the skills or time necessary to use or extend the original C implementation. In this article, 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 features include visualization tools, powerful scripting, built-in data graphing, adjustable levels of external and internal noise, and adjustable lexical characteristics, such as frequency of occurrence. Functions for saving and reloading entire simulations facilitate archiving, sharing, and replication and also make jTRACE ideal for educational use, since it comes bundled with several important simulations. jTRACE can be downloaded from magnuson.psy.uconn.edu/jtrace.

TRACE is arguably the most successful model of spoken word recognition (SWR) to date. However, although it is widely discussed, it is not widely used. One obstacle is that the original implementation in the C programming language is opaque to the average psychologist. We present jTRACE, a user-friendly cross-platform and free software tool that reimplements the TRACE model in the Java programming language. jTRACE accommodates the needs of both researchers and students, allowing beginners to ignore many details of simulation parameters, while giving powerful scripting tools to advanced users.

This article is structured as follows: This introduction will outline our motivations for creating jTRACE. The next section will give a primer on how TRACE works and also will introduce the graphical user interface of jTRACE. The next sections will describe the principal functions that make jTRACE an effective and versatile tool and will walk through some useful examples. The final section will review some key simulations we have replicated in jTRACE and will describe a validation metric used to verify that jTRACE simulations are virtually identical to simulations performed with the original TRACE implementation. Readers are encouraged to download jTRACE from magnuson.psy.uconn.edu/jtrace and to explore the software as they read.

Why jTRACE?

In their seminal article, McClelland and Elman (1986) introduced TRACE and described an array of simulations of human-speech-processing tasks. McClelland and Elman started from the core principles of the Cohort model (Marslen-Wilson & Tyler, 1980) and the interactive activation model of letter and visual word perception (McClelland & Rumelhart, 1981) to create an implemented model of speech perception and SWR. Although TRACE incorporates key characteristics of spoken word processing first worked out in the Cohort model, it is a distinct theory, in that it abandoned the rule-based nature of the original Cohort model. Instead, there is parallel, graded activation of features, phonemes, and words on the basis of fine-grained similarity to the input. Interactive-activation mechanisms, such as bidirectional influences between units on different processing levels and lateral inhibition between units within a layer, allow TRACE to activate multiple phoneme and word candidates and, for appropriate lengths of input, eventually select one (in

J. S. Magnuson, james.magnuson@uconn.edu

cases in which the input is an unambiguous phoneme or word), although there is no built-in moment of recognition. Instead, phoneme and word recognition emerge as series of activation peaks.

The fact that TRACE was fully implemented as a computational model represented a significant advance in the specificity of speech perception and word recognition models. An implementation requires precise specification of mechanisms that correspond to elements of a theory and forces the implementer to grapple with difficult input, output, and processing issues that might not be apparent from the vantage point of an unimplemented model, and an implemented model generates detailed, falsifiable predictions about human behavior. In the original article, TRACE was used to account for a variety of speech perception and SWR phenomena, including categorical perception of phonemes, word segmentation, lexical effects on phoneme perception, and perceptual restoration given degraded input.

Although TRACE was introduced 20 years ago, it continues to be vital in current work in speech perception and SWR. Despite well-known limitations (acknowledged in the original 1986 article and discussed below), TRACE is still the best available model, with the broadest and deepest coverage of the literature and, arguably, with the most realistic input representation of any current model. Protopapas (1999) provides a detailed discussion of TRACE's place in speech perception research.

TRACE has proved extremely flexible and continues to spur new research and provide a means for theory testing. For example, it has provided remarkably good fits to eyetracking data from recent studies of the time course of lexical activation and competition (Allopenna, Magnuson, & Tanenhaus, 1998; Dahan, Magnuson, & Tanenhaus, 2001), including subtle effects of subphonemic stimulus manipulations (Dahan, Magnuson, Tanenhaus, & Hogan, 2001). As the prime example of an interactive model, TRACE has been of notable importance in an ongoing debate in the SWR literature between proponents of purely feedforward (autonomous) models (e.g., Norris, McQueen, & Cutler, 2000) and proponents of interaction (Elman & McClelland, 1988; Magnuson, McMurray, Tanenhaus, & Aslin, 2003; Samuel & Pitt, 2003). Thus, TRACE continues to hold a central position in models of speech perception and SWR.

However, TRACE is actually used much less than one might expect. Instead, although many articles in speech perception and SWR discuss TRACE, there is a tendency for researchers to make inferences about what TRACE would predict on the basis of logical expectations about how TRACE should perform on a particular task. Researchers then work from these predictions without confirming them via simulation, and in some cases, the intuitions turn out to be wrong (for examples in which TRACE behaves quite differently than previous researchers had expected it would,[1] see Dahan, Magnuson, & Tanenhaus, 2001; Dahan, Magnuson, Tanenhaus, & Hogan, 2001; Magnuson, Dahan, & Tanenhaus, 2001; Mirman, McClelland, & Holt, 2005).

We expect that researchers are reluctant to do actual TRACE simulations because TRACE is difficult to use. McClelland and Elman (1986) originally implemented TRACE as a C program (which we will call *cTRACE* when referring to implementation-specific details) that 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 must be analyzed separately. On contemporary desktop computers, cTRACE performs single simulations in a few seconds, is straightforward to use for users with a modicum of UNIX and/or programming expertise, and has contributed to many publications. However, actively using cTRACE can be daunting, because learning to use the software and determining how to process its output require substantial time and effort.

Furthermore, cTRACE adheres to 1980s programming standards, but standards have changed substantially (including improvements in conventions to promote shareability and readability). The code is complex and somewhat difficult to extend, although extensions to the original model have been undertaken to fit SWR data previously unaccounted for by TRACE. For example, Dahan, Magnuson, and Tanenhaus (2001) tested three different implementations of lexical frequency in TRACE. Although such extensions may consist of just a few lines of code, correctly placing those lines requires understanding many details of the original cTRACE code, which, in turn, requires hours of study.

These ease-of-use and extension issues were our primary motivation for reimplementing TRACE. One goal was to develop a platform-independent and user-friendly TRACE program that includes visualization tools, in order to encourage wider use of the model. A corollary goal was to develop a *programmer*-friendly implementation that would encourage researchers to extend the TRACE model in new ways. The result is jTRACE, a reimplementation of TRACE in Java, which can be run on any modern desktop computing platform (UNIX, Linux, Macintosh OS X, or Windows). jTRACE improves upon the original with an easy-to-use graphical user interface (GUI) that allows users to set up and run TRACE simulations in minutes without any knowledge of programming. Once set up, simulation parameters and results can be saved to simple output files (in XML format) and shared with colleagues, facilitating replications and follow-up studies.

How TRACE Works

The TRACE model is a connectionist network with an input layer and three processing layers: pseudospectra (feature), phoneme, and word. Figure 1 shows a schematic diagram of TRACE. There are three types of connectivity: (1) feedforward excitatory connections from input to features, features to phonemes, and phonemes to words; (2) lateral (i.e., within-layer) inhibitory connections at the feature, phoneme, and word layers; and (3) top-down feedback excitatory connections from words to phonemes and from phonemes to features (although the default setting of phoneme–feature feedback is 0.0).

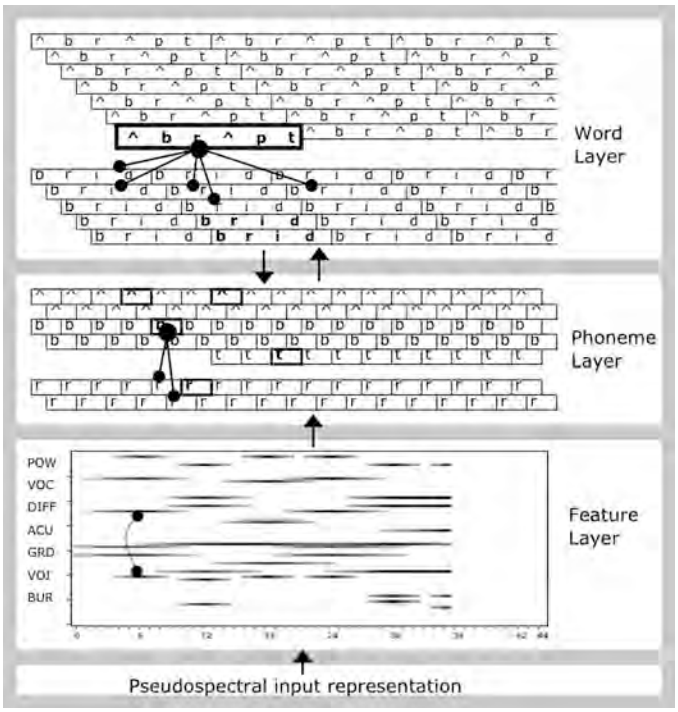

Figure 1. Schematic of TRACE's architecture. Each of TRACE's layers is fully connected to the subsequent layer, leading to thousands of connections. Unidirectional arrows denote feedforward and feedback excitation. Lines with circles on the ends indicate within-layer inhibition (a tiny subset of connections is shown). Note that although feedback from phonemes to features is possible, it is typically set to 0.0.

An external stimulus is passed to the input layer, and each processing cycle sends activity along the connections, changing the activation values in the processing layers. Parameters govern the strength of the excitatory and inhibitory connections, as well as many other processing details. However, in most simulations, most or all parameters are left at their default values.[2]

Input and feature layers. The input to TRACE is a pseudospectral representation based on acoustic-phonetic features (McClelland & Elman, 1986). The input takes the form of a 63-dimensional vector for each time step. The 63-dimensional vector describes the activation of seven acoustic features, each consisting of nine steps in a continuum (e.g., from fully voiceless to fully voiced). TRACE includes a text-to-"speech" (pseudospectral representation) function. The user can type in a string of phonemes from TRACE's inventory (/p/, /b/, /t/, /d/, /k/, /g/, /s/, /ʃ/, /r/, /l/, /a/, /i/, /u/, /ʌ/, and the silence phoneme, /-/), and TRACE generates the input representation.

The bottom left window in Figure 2 shows jTRACE's representation of 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 magnitude of activation. The stimulus in this example is the phoneme sequence /-ʌbrʌpt-/, which stands for the English word *abrupt* with a short silence at both ends. Each input vector in TRACE is meant to correspond to approximately 10 msec of real time,[3] and input vectors are applied in succession one time slice per cycle.

Note that although an input vector is applied for just one time cycle, its impact persists on the feature layer, due to slow *decay*. The activation of a feature unit is the sum of (1) its bottom-up input, (2) its top-down input (if phoneme–feature feedback is on), (3) negative inputs via lateral inhibition from other feature units, and (4) its activation at the previous time step, multiplied by a constant decay parameter. The fact that feature units remain active after an input is applied until they decay back to resting levels means that they provide a semipersistent input to the phoneme layer.

Phoneme and word layers. The activation of a phoneme unit at a given time step is the sum of bottom-up input from features, top-down feedback from words, negative input from other phoneme units via lateral inhibitory connections, and the unit's activation at the previous time step, attenuated by a constant proportional decay. Similarly, the activation of a word unit is the sum of bottom-up input from phonemes, negative input from other word units via lateral inhibitory connections, and the unit's activation at the previous time step, attenuated by a constant proportional decay.

The phoneme and word layers have a somewhat complex organization, vis-à-vis number and connectivity of units, due to TRACE's twofold representation of time. The passage of real time is represented by the successive processing cycles

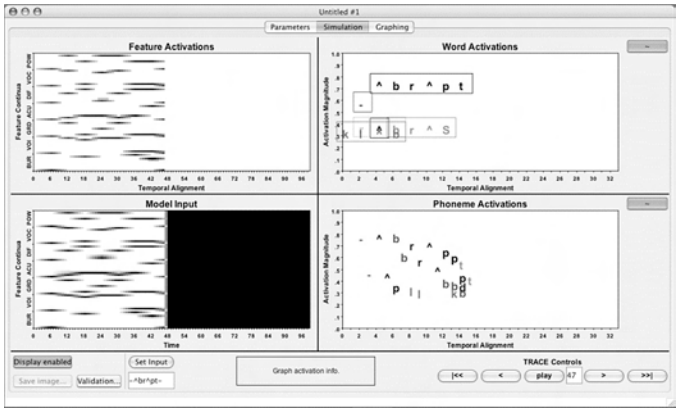

Figure 2. The jTRACE simulation panel.

that cumulatively reveal more of the external stimulus. As a visual aid, the future input in jTRACE is completely obscured at the beginning of a simulation, and its representation is revealed cycle by cycle (see Figure 2, lower left).

In addition to real time, TRACE represents the temporal alignment of the words and phonemes that become activated in memory. In the phoneme activation window of Figure 2 (lower right), activations are plotted using the *floating unit* representation used for some figures in McClelland and Elman (1986). The temporal alignment of each activated phoneme (relative to the start of the utterance) is indicated by horizontal position, whereas the strength of its activation is indicated by vertical position. This representation provides temporal extent; that is, it permits TRACE to process sequences of acoustic-phonetic events, clearly a requirement for speech perception. The representation of time slices also permits multiple phonemes to become activated at the same temporal alignment (see the middle of the phoneme window, where many phonemes are about equally activated and none are strongly activated, indicating substantial uncertainty about the signal at that point).

As with phoneme units, each word unit has a temporal alignment and activation strength and can be represented visually as floating units—horizontal alignment and vertical strength. The word layer's temporal representation permits TRACE to perceive sequences of words arranged in time, a requirement for segmentation of continuous speech into words. Multiple words can be activated at overlapping temporal alignment, and this again is interpreted as uncertainty. In the word activation window in Figure 2 (upper right), the word *abrupt* is most active, but the words *agree* and *blood* remain active at overlapping alignments.

When TRACE processes /-ʌbrʌpt-/, it will "hear" /ʌ/ occurring once early on and again later in the word. The phoneme layer has multiple /ʌ/ units distributed evenly across the temporal alignments, allowing TRACE to recognize occurrences of a single phoneme at multiple temporal positions. The same principle holds at the word layer, allowing TRACE to recognize multiple instances of a word, as in *dog chases dog*.

Another consequence of TRACE's temporal representation is that a word unit can become activated on the basis of partial bottom-up information. In combination with lexical-to-phoneme feedback and lateral inhibition, this capacity allows TRACE to model, for example, uniqueness point effects (Frauenfelder, Segui, & Dijkstra, 1990; see Frauenfelder & Peeters, 1998, for an examination of uniqueness points in TRACE). As a word is heard, lexical feedback is passed to all of its constituent phonemes. Thus, when only the first few sounds of *elephant* have been presented, later phonemes become somewhat active from feedback. Because of lateral inhibition, the fewer competitors a word has, the more quickly it will be activated. Feedback and inhibition will conspire to generate faster activation for a word that has an early uniqueness point (with competitors overlapping only in the first couple of phonemes) than for a word that has a late uniqueness point (with competitors overlapping almost or completely to its end).

Thus, unit reduplication lays out, in an explicit, spatial fashion, the contingencies between activations of units at feature, phoneme, and word levels in a way that keeps track of the temporal relationships of units within and between layers. This architecture is necessary if TRACE is to recognize sequences of phonemes and words, as opposed to single words aligned with a slot representation (as in the interactive activation model of McClelland & Rumelhart, 1981, among other models). It also allows TRACE to solve the segmentation problem (the fact that there are no invariant cues to word boundaries in real speech; see, e.g., Miller & Eimas, 1995). There is no explicit search for word boundaries in TRACE. Rather, activation and inhibition result in a series of phoneme and word units that temporarily win the competition for "recognition" at different points in time.

However, unit reduplication provides a somewhat inelegant solution to the temporal representation and segmentation problems (as McClelland & Elman discussed in 1986) and has been the target of substantial criticism. Norris (1994) cites this "highly implausible architecture" as a failing great enough to motivate an alternative model, Shortlist. Although claims about the number of nodes needed to implement such a scheme tend to overestimate a bit,[4] the number unit reduplication was intended as an

Table 1
Simulations Conducted in cTRACE and jTRACE in Order to Validate jTRACE

Effect and Publication Source	Max SMAD	No. of Simulations
1. Lexical effect on phoneme perception (McClelland & Elman, 1986, p. 24)	0.0016	9
2. Elimination of lexical effects by time pressure (*Ibid.*, p. 26)	0.0009	2
3. Late lexical effects (*Ibid.*, p. 27)	0.0023	6
4. Dependence of lexical effects on phonological ambiguity (*Ibid.*, p. 28)	0.0012	2
5. Absence of lexical effects in some reaction time studies (*Ibid.*, p. 30)	0.0031	2
6. Lexical effects in reaction time studies (*Ibid.*, p. 29)	0.0036	4
7. "Lexical conspiracy" effect (*Ibid.*, p. 33)	0.0022	4
8. Time course of word recognition effects (*Ibid.*, p. 57)	0.0022	2
9. Lexical basis for word segmentation (*Ibid.*, p. 63)	0.0017	4
10. Recognizing words in short sentences (*Ibid.*, p. 69)	0.0067	3
11. Recognition of all items in the SLEX lexicon (*Ibid.*, p. 62)	0.0056	211
12. Nonwords are difficult to segment, but lexical activation facilitates segmentation by offering a cue to word boundaries (*Ibid.*, p. 65)	0.0011	2
13. Lexical basis for word segmentation, part 2: segmentation of word pairs (*Ibid.*, p. 65)	n/a	211
14. Phoneme context effects in the stochastic version of cTRACE (McClelland, 1991, Figure 9, p. 25)	n/a	4,500
15. Time course of frequency effects in eyetracking experiment (Dahan, Magnuson, & Tanenhaus, 2001, Figure 6, p. 342)	n/a	68
16. Evaluating the contribution of lexical feedback on word recognition (Frauenfelder & Peeters, 1998, Figure 4.9, p. 139)	n/a	900

abstract characterization of the process, rather than as a proposal about its actual neural implementation, which is likely to involve distributed, rather than localist, representations and a less explicit representation of units such as phonemes and words (McClelland, Mirman, & Holt, 2006).

New Features in jTRACE

We have added many useful new features to TRACE, in addition to the GUI. This section will describe 11 of jTRACE's functions that make it a powerful, versatile, and user-friendly program. These functions have greatly facilitated the replications described in Table 1 and are currently leading to the development of new TRACE modeling results.

1. *Graphical user interface*. One obstacle to widespread use of cTRACE is the user interface. Although straightforward for the initiated, the text-based interface can be daunting. The GUI in jTRACE makes the program both more approachable and more powerful. In jTRACE, multiple documents exist within self-contained windows, and a menu bar offers such functions as saving, loading, screen layout, and help documentation. Figure 3 is a screenshot showing multiple windows open in jTRACE.

Each jTRACE simulation "document" consists of three tabs. The *parameters tab* includes a tabular interface for modifying simulation parameters. The *simulation tab*, pictured in Figure 2, offers a real-time visual representation of TRACE's activation levels. The *graphing tab* permits the creation of graphs containing activation values or re-

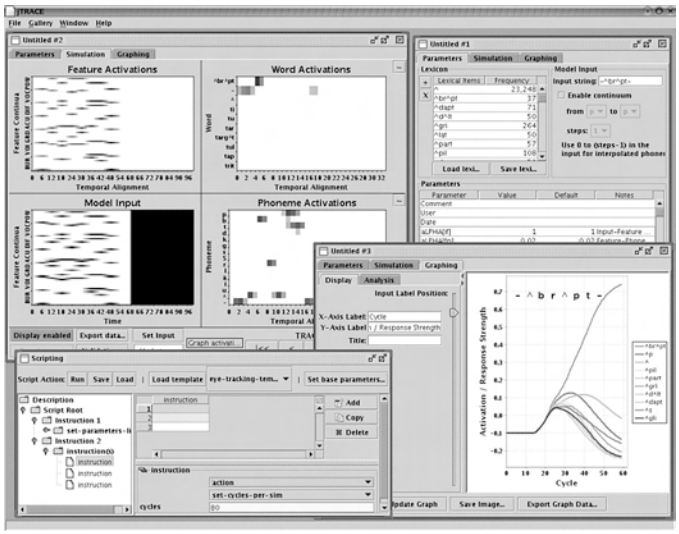

Figure 3. Multiple windows in jTRACE.

sponse probabilities of words and phonemes. The user can open multiple documents at once, each with its own tabs. In addition, a scripting window allows sets of simulations to be run in a batch mode, with the results exported to files for further analysis or displayed within jTRACE.

2. *Visualization*. The simulation and graphing tabs provide intuitive visualization tools. In the simulation tab, word activations and phoneme activations can be visualized as floating units wherein a unit's vertical height corresponds to the magnitude of its activation. This type of visualization is based on diagrams used in the original TRACE article. As the simulation progresses in real time, these floating units are animated, showing activation values changing over time. The most active units are automatically selected for inclusion in the floating unit graph.

Input, feature, phoneme, and lexical layers can also be visualized with a spectrogram-like grayscale matrix format, and the input and feature layers are always displayed using that format. Figure 4 shows floating unit (left) and corresponding matrix (right) representations of word and phoneme units. In the feature graphs, each cell corresponds to a single unit aligned with that point in time. In the phoneme and word matrix graphs, each cell is aligned with the onset of a phoneme or word unit. However, the cells are only one time unit wide, despite the fact that the units they represent have temporal extent of at least three time steps (see Figure 1). But by using only one cell per unit and aligning them with unit onsets, all the units for a particular phoneme or word can be displayed in a single row in the matrix. This provides a compact method for examining the activations of entire layers, as opposed to a subset of highly active items. In this format, the *x*-axis represents the temporal alignment of the units, the *y*-axis represents the individual feature, phoneme, or word units being represented, and the darkness of the cells represents the strength of the activation. Using the control buttons (play, rewind, etc.), the matrix representation changes with the developing simulation. By dragging the mouse over individual cells, their exact numerical activations are shown in a box below.

The graphing panel, pictured in Figure 5, generates graphs of phoneme or word unit activations over time. In addition, users may plot response probabilities using the Luce (1959) choice rule. Several options for calculating this measure, based on earlier work (Allopenna et al., 1998; Frauenfelder & Peeters, 1998; McClelland & Elman, 1986), 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.

3. *Tabbing and cloning*. The main window for any simulation has three tabs: parameters, simulation, and graphing. Multiple simulations may be opened simultaneously, since the jTRACE GUI uses a *multiple document interface* approach to manage multiple simulations as separate "documents" (see Figure 3 for a screenshot illustrating multiple simultaneous simulations), which can be moved by hand or arranged automatically. Another feature this enables is the ability to "clone" a simulation document. Choosing "Clone" from the "File" menu creates a complete copy of the current simulation, including all parameter and graphing settings. This is particularly useful if you want to make a change to the input or some parameter and then compare those results with your current results.

4. *Scripting*. The scripting panel (Figure 6) automates simulation preparation, execution, and analysis. Groups of simulations can be performed iteratively over lexical items, ranges of parameter values, input values, phoneme continua, or even analysis settings. A script consists of a hierarchical organization of the following types of expressions. An *iterator* is an instruction on how to change a particular parameter value at each successive simulation. A *query* fetches some information about a simulation and optionally processes it (an example of a query is a decision rule, described below). An *action* is an instruction to save data, run a simulation, change a parameter value, or some other simple operation. A *conditional* asks a question about a simulation and directs action on the basis of the response to that question (like an if–else statement).

jTRACE visualizes scripts as a tree of expressions. Branches and leaves can be added or removed, and the details of their processing are specified via a simple interface. A number of scripting templates are included, so users never have to start a script from scratch. Each tem-

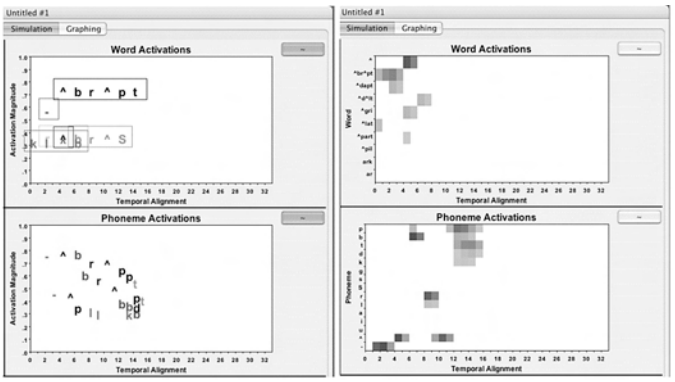

Figure 4. Comparison of floating unit (left panel) and corresponding matrix representations (right panel) of word and phoneme units. Clicking on the "tilde" buttons toggles between representations.

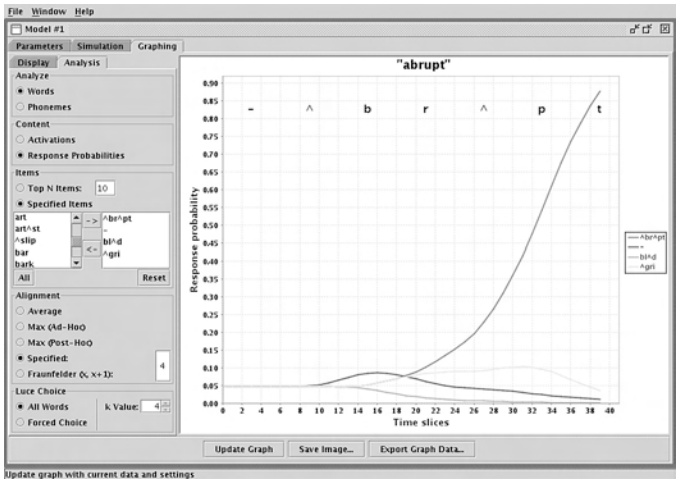

Figure 5. The graphing panel.

plate takes advantage of a few of the most useful scripting tools to accomplish a specific modeling goal. The templates are useful both for learning about scripting functions and as a resource for designing complex TRACE studies. Scripts can be saved to an XML format. Saved scripts can be run from the command line without invoking the GUI, in order to speed processing. Advanced users may find it useful to write or modify scripts directly in XML using a text or XML editor, rather than the GUI. The jTRACE user manual contains complete instructions on how to use scripting.

5. *Decision rules*. Several decision rules for interpreting the lexical and phoneme layers of TRACE are implemented in the scripting panel on the basis of rules used by McClelland and Elman (1986), McClelland (1991), Frauenfelder and Peeters (1998), and recent studies comparing TRACE with eyetracking data (Allopenna et al., 1998; Dahan, Magnuson, & Tanenhaus, 2001). A decision rule is a linking hypothesis between word and phoneme activations in TRACE and human data. TRACE decision rules must consider (1) which units to include as competitors (ranging from only those units that are aligned with the target, as in earlier studies, or all units, as in more recent ones); (2) how to operationalize "recognition" in the case in which the model is used to account for tasks such as lexical decision, or how to map activations onto fixation proportions over time, in the case of eyetracking; and (3) whether to transform TRACE activations and how. The most common type of scripting used so far with jTRACE is one that iterates over lexical items and applies a decision rule to test for lexical recognition under a given set of simulation parameters.

6. *Parameter extensions*. Two important extensions to the original TRACE model have been implemented in jTRACE. The first is *Gaussian noise* that can be applied to any layer (directly to input values, or to any layer of processing units). Input noise (external noise) allows simulation of speech perception in noisy conditions, whereas noise added to processing layers (internal noise) makes TRACE stochastic, as per McClelland (1991).

Second, the three implementations of lexical frequency described by Dahan, Magnuson, and Tanenhaus (2001) have been implemented. These are *resting level* (baseline activations are proportional to word frequency), *postactivation* (the computation of response probabilities includes frequency values, which means that frequency does not affect the dynamics of lexical activation and competition directly), and *connection strengths* (phoneme-to-lexical weights are proportional to word frequency, so that the connection from /k/ to *cat* would be stronger than that from /k/ to *cad*).

7. *Save/load features*. All the details of a jTRACE simulation, including parameters, lexicons, raw activation values, graph data, decision rule results, and scripts, can be saved to external files of different formats. Choosing to save files to the .jt format allows them to be reloaded into jTRACE later on. We expect these files will be useful for sharing among collaborators, for archiving simulations in a standard format, and for educational use. Data can also be saved to a comma-separated text format, which can easily be loaded into another application for analysis. Finally, snapshots of graphs and simulation windows can be saved to graphics files.

8. *Simulation gallery*. Since simulations can be saved and reloaded, we put together a *gallery* of saved classic simulations (from the validation project described below). You simply choose from a list of simulations in the "Gallery" menu. The simulation is loaded, complete with appropriate parameter settings and even graphing windows, and so forth, tailored to the simulation. Users can add to the gallery by putting their own .jt files in the gallery directory. So if one were to use jTRACE for a course lab session, it would be easy to make a gallery of simulations for students to run or a gallery of incomplete simulation setups that students could be asked to complete.

9. *Modernized code and data formats*. jTRACE has been coded with contemporary programming practices and is thoroughly commented. This will facilitate extensions and exploration of how the code works. We also devoted considerable effort to keeping the code flexible

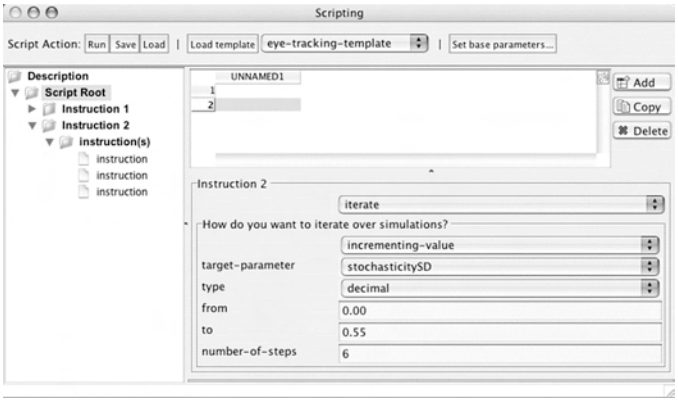

Figure 6. An example of the scripting interface.

and extensible, in order to allow others to modify existing functions or add new ones. To this end, we have written a programmer's guide, which is available along with the source code from the jTRACE Web site. Data output is now marked up in XML format, making it much more transparent and archivable. The code and data formats conform to current standards (for Java and XML).

10. *Platform independence*. Because jTRACE was implemented in Java, the program is completely platform independent (the necessary *Java virtual machine*, or JVM, is commonly installed by default when Internet browsers are installed). We have tested jTRACE on Macintosh, Linux, and Windows platforms.

11. *Validation*. jTRACE has the ability to load simulation data generated by cTRACE. This allows easy comparison between the two models for any simulation. A researcher with doubts about the accuracy of jTRACE as a reimplementation can perform his or her own validation comparisons. This function requires some technical skill, however, since one must install and compile an augmented version of cTRACE that provides output compatible with jTRACE. These tools (which we used for the large-scale validation project described below) are available from our Web site.

Together, these 11 new features make jTRACE simultaneously useful for novice and expert modelers. Novices (whether students or researchers) can be doing their own simple" simulations within a few minutes of starting the program or complex batch simulations in less than an hour. The standard Java and XML practices we followed in developing jTRACE make it ready for advanced users and programmers to extend. To illustrate how easy jTRACE is to use, the next section will walk through an example of running an actual simulation.

A jTRACE Example: Lexical Effects in a Reaction Time Study

To illustrate some of the functions discussed so far, we offer a walk-through of two simulations from Replication 7 (see Table 1). McClelland and Elman (1986, p. 33) suggested that phonotactic effects on phoneme perception may be the result of a "conspiracy" of lexical feedback activation acting upon an ambiguous phoneme unit, thus causing perception to be biased in favor of the lexical evidence. To investigate the role of lexical feedback in phoneme perception, McClelland and Elman created an ambiguous phoneme, /?/, halfway between /l/ and /r/. They placed this phoneme within a lexical context that favors the /r/ interpretation (/t?i/) and another that favors the /l/ interpretation (/s?i/). These contexts are biased in the sense that in the TRACE lexicon, as in English, /sli/ and /tri/ occur word initially, but /sri/ and /tli/ do not. Given these items, TRACE produces the expected phonotactic effect, restoring /?/ as /l/ in one case and as /r/ in the other. The researchers point out that the context effect in TRACE is instantiated by lexical feedback connections; phoneme representations do not explicitly encode any positional probability values.

To replicate this result in jTRACE, follow these steps.

1. Start jTRACE.
2. From the menu bar, choose *File:New model*.
3. In the upper right section of the parameters tab of the new document:
   a. Check "enable continuum."
   b. Set *from* to l and *to* to r.
   c. Set steps to 3.
   d. Set the input string to: -s?i-.
4. Click on the simulation tab.
5. Click "play" at the bottom right, let the simulation run for about 60 cycles, and then click "stop."
6. Click on the graphing tab.
7. Select the analysis tab within the graphing tab, and select the following settings:
   a. Analyze: phonemes
   b. Content: activations
   c. Items: specified items
   d. Select l and r from the unselected list and move them to the selected list by clicking the arrowbutton.
   e. Alignment: Either select "specified" and enter 6,or choose "Max post-hoc," which will automatically determine that 6 is the best alignment.

8. Click "update graph." Note that the /l/ and /r/ activations are identical for about 25 cycles, and then begin to diverge as lexical feedback begins to have a significant impact.
9. From the menu bar, choose *File:Clone model*.
10. In the parameters tab of the new simulation document, set the input string to "-t?l-."
11. Click on the simulation tab.
12. Click "play" at the bottom right, let the simulation run for about 60 cycles, and then click "stop."
13. Click on the graphing tab. All the analysis settings done in Step 7 are already in place. The results are very similar, except that in this case, the lexical feed back was stronger and /r/ diverges from /l/ a couple cycles earlier than /l/ diverged from /r/ in the previous simulation.
14. To compare the two simulations, from the menu bar choose *Window:Tile*, so that the two graphing windows may be examined side by side.

Following the steps in this example demonstrates some of the features of jTRACE. To learn about other features, please see the jTRACE user manual (from the Help menu), or just explore the application.

Applications of jTRACE

Research. jTRACE can, of course, be put to all the same uses as cTRACE. The visualization tools allow greater insight into the workings of the model online, as simulations are carried out. Scripting facilitates conducting complex batches of simulations. Although one could run batches with cTRACE (simply by creating a text file containing every command needed—e.g., to repeat a simulation with every word in the lexicon), changing certain parameters required recompiling the software;[5] jTRACE allows you to iterate over any parameter in specified step sizes, greatly facilitating explorations of parameters spaces and so forth.

Education. Although the new features of jTRACE make it extremely powerful for research, it is also easy enough to use for inclusion in course labs. Students can run the "canned" examples from the gallery and report on them. Instructors might, instead, create a simulation and save it in the gallery but leave a few crucial steps (parameter settings, etc.) for students to complete. In a computational modeling course or a psycholinguistics course, students could use jTRACE to design their own simulations to explore aspects of speech perception, word recognition, or the model itself (e.g., the role of feedback).

Validating jTRACE

This section will summarize a number of TRACE simulation replications that we have performed using jTRACE. Each of these simulations was originally done using cTRACE. We selected simulations for replication with the goal of assembling a set of simulation replications that are representative of the types of effects TRACE has been used to study previously. These effects include phoneme and lexical tasks, fitting time course data from eyetracking studies, and McClelland's (1991) stochastic version of cTRACE. Table 1 presents the list of simulations chosen for validating jTRACE.

Simulations 1–13 were drawn from the original TRACE article and represent the core evidence used by McClelland and Elman (1986) to argue for the interactive activation framework. The parameter and stimulus sets for each of these simulations are bundled with the jTRACE download file and can be loaded from the "Gallery" menu. Simulations 14–16 were based on three studies that extended a particular aspect of the TRACE model in order to evaluate complex phenomena in speech perception and word recognition.

Simulation 14 was from McClelland (1991). Massaro (1989) criticized interactive models because they fail to account for evidence of independent combination of bottom-up and top-down information sources. McClelland hypothesized that this was due to the absence of any direct analogue to variability in interactive models such as TRACE (rather, deterministic interactive models can be viewed as models of central tendencies). He therefore developed a stochastic version of TRACE (one in which noise can be applied to the input and/or processing units, allowing its response to the same stimulus to vary from simulation to simulation). By running a series of simulations with the stochastic TRACE model, McClelland found that it correctly predicted independent stimulus and context effects. Simulation 14 replicated the stochastic simulations in jTRACE to arrive at the same result.

Simulation 15 was based on three lexical frequency extensions explored by Dahan, Magnuson and Tanenhaus (2001), who conducted eyetracking 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 the earliest moments of processing, they compared TRACE fits with three different implementations of frequency: (1) phoneme-to-word connection strengths proportional to word frequency, (2) word-resting levels proportional to word frequency, and (3) frequency incorporated into a postactivation decision rule. All three are implemented in jTRACE.

The final simulation listed in Table 1 is not an exact replication. Simulation 16 comes from a recent project in which we reexamined Frauenfelder and Peeters's (1998) investigation of lexical feedback. Frauenfelder and Peeters (Simulation 6, p. 139) reported that for half of the 21 words they examined, turning off lexical feedback speeded recognition time. This finding has been used to bolster the case against interactive models, since it implies that lexical feedback serves no useful function (Norris et al., 2000).

We replicated the main finding of that simulation. With a similar lexicon (the original was unavailable), about half of the items with characteristics like those used in the original simulation were recognized more quickly without feedback. However, when we tested all the words in the 900-word lexicon, we found that the majority (73%) were recognized more quickly with feedback than without. Furthermore, when we tested recognition of every lexical item with in-

creasing levels of noise added to inputs, feedback promoted greater accuracy and faster recognition. For details about this work, see Magnuson, Strauss, and Harris (2005).

Validating replications against the originals. We used two methods to validate jTRACE simulations against their cTRACE equivalents. The first method applies to Simulations 1–12. 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). This value between 0 and 100 summarizes the amount of 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).

The SMAD metric is computed as follows:

$$\text{SMAD} = \frac{100}{\| v \|} \sum \frac{| v_c - v_j |}{\max - \min}, \quad (1)$$

where $\|v\|$ is the number of elements in the four-dimensional simulation arrays, v_c and v_j refer to the iteration over every corresponding cell pair in the cTRACE and jTRACE simulation arrays, and max and min are the maximum and minimum unit activation, consistently set to 1.0 and −0.3. The metric can thus be interpreted as a percentage of possible error.

The SMAD metric was applied to all of the simulations in Replications 1–12, amounting to more than 250 individual simulations. The SMAD never exceeded 0.007% for any of those simulations; the average SMAD score was 0.0018%. Table 1 states the largest SMAD score obtained for each replication. The largest individual unit difference (the absolute difference between a particular pair of unit activations from jTRACE and cTRACE) was below 3%, although the majority of the simulation comparisons had maximum unit differences below 1%. We conclude that the difference between each simulation pair—less than 0.007%—is small enough to claim that a replication has been performed. These minor differences stem from some unavoidable algorithmic changes (e.g., cTRACE relies heavily on pointer arithmetic, which is not available in Java), as well as on numerical precision and rounding differences.

Figure 7 illustrates how we validated SMAD as a comparison metric. One might worry that SMAD scores would generally be low, because many units' activations will be low at every time step in a simulation. To provide an indication of how meaningful our low SMAD scores are, we sought to establish the range of meaningful SMAD scores by comparing simulations using different words—which should result in substantially higher SMAD scores than would cTRACE/jTRACE comparisons. We ran simulations with every word in the 213-word SLEX lexicon (the original TRACE lexicon used by McClelland & Elman, 1986) and computed SMAD scores for all word pairs (a total of 45,369 scores). Comparing a word with itself always resulted in a score of zero. The SMAD scores computed from the remaining comparisons were enumerated, and the distribution was plotted. Figure 7 shows that the maximum error between jTRACE and cTRACE simulations is indistinguishable from zero when compared to the differences between distinct simulations.

The second method for validating replications was to replicate the published figures, or published results, that summarize groups of cTRACE simulations.[6] This method applies to simulations for which direct comparison is impossible and/or to modeling efforts that involve very large batches of simulations. This method was used for Replications 13–16. We will briefly review the results for each here.

Replication 13. McClelland and Elman (1986) randomly generated 211 word pairs. This created inputs such as /-lustriti-/ (*loose treaty*) and /-trʌbʌlrak-/ (*trouble rock*). Each of these pairs was simulated in TRACE. A successful parse occurred only if the two most active word units during the simulation were the same words as those used to create the pair. TRACE successfully parsed 189 of 211 pairs (90%), and most errors were due to multiple possible parses or coarticulatory effects. Although the original stimuli were not available, we generated a new set of 211 pairs. In our simulations, 194 of 211 (94%) pairs were successfully segmented, and errors fell into the same categories as in the original simulation.

Replication 14. Although McClelland (1991) determined that multiple variants of stochastic interactive models could generate independent integrations of stimulus and context, we focused on the intrinsic noise version he used to generate his Figure 9. McClelland originally ran 4,500 simulations (using a case like McClelland & Elman's [1986] *lexical conspiracy* simulations [Simulation 7 in Table 1 in this article]) with the stochastic version of TRACE and found that it correctly predicted independent stimulus and context effects (as evidenced by parallel effects for each context when the results are transformed to *z* scores; McClelland, 1991, Figure 9; see Movellan & McClelland, 2001, for further discussion). We followed McClelland's report to change various parameters[7] and some phoneme

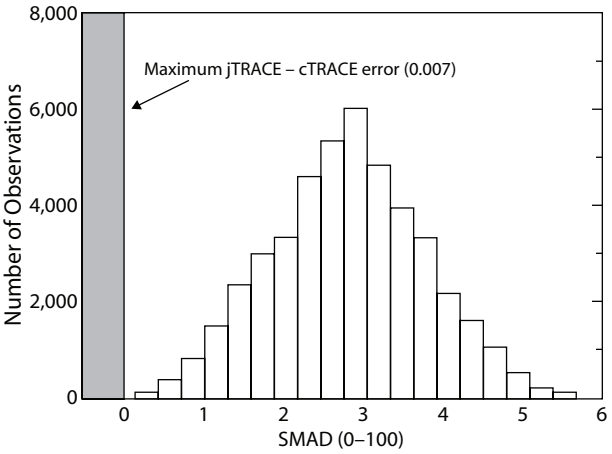

Figure 7. The distribution of scaled mean absolute difference (SMAD) scores (see the text) for comparisons of every word in a 213-word lexicon to every other word in the lexicon. The maximum difference between cTRACE and jTRACE (0.007) is much closer to zero than is the lower end of the SMAD distribution.

definitions, and reran all of the simulations in jTRACE. We replicated the crucial independence of context and stimulus, although our lexical effects were slightly larger than those of McClelland. The average root-mean square (RMS) error between jTRACE and the original McClelland simulation was about 0.16. This is a relatively large difference (repeated replications with the original code yield an average RMS error of about 0.08). Some of the difference can be attributed to precision differences, but not all of it. A closer replication would require line-by-line comparison of jTRACE and the modified version of cTRACE McClelland developed specifically for this project. Since the current version of jTRACE provides a close fit to every other cTRACE simulation we replicated, and it provides the correct qualitative behavior when tested on the McClelland (1991) simulations and, arguably, a reasonably close quantitative fit, we did not pursue this further.

Replication 15. Following the methods of Dahan, Magnuson, and Tanenhaus (2001), we implemented their three frequency mechanisms in jTRACE (resting levels, bottom-up connection strengths, and postperceptual bias). Using the same lexicon and items, we ran the 68 simulations required for replication of the results shown in their Figure 6. jTRACE provided a very close replication, with RMS error averaging approximately 0.01 for the three mechanisms.

Replication 16. As was mentioned above, an exact replication of Frauenfelder and Peeters's (1998) result was not possible, because the original lexicon and input set were unavailable. When we restricted our analysis to words in our large lexicon with the same properties as Frauenfelder and Peeters's (seven phonemes long, uniqueness point at position four), we found that there was no general advantage for feedback (see Magnuson et al., 2005). When we went beyond their simulations and tested all the words in the lexicon, we found that feedback in TRACE speeds processing and makes the model robust in noise.

Summary. Strict validation methods were used to ensure that replications of previous results were accurate quantitatively, as well as qualitatively. Furthermore, the number and type of replications performed provides a broad coverage of the effects TRACE has been used to model. Thus, jTRACE is a faithful reimplementation of cTRACE.

Conclusions

The platform-independent tool jTRACE should be of great use to researchers in speech perception and word recognition. The command line user interface of the original cTRACE has deterred its widespread use for basic modeling tasks. Several features of the cTRACE implementation have deterred extensions to the model. jTRACE resolves these two issues by focusing attention on usability and up-to-date programming conventions. Concerns about the precision of jTRACE as a reimplementation of the original model have been addressed by an extensive validation effort.

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. jTRACE offers a framework wherein diverse types of extensions to the original model can be implemented and combined with one another, using simple parameters.

AUTHOR NOTE

Development of jTRACE and preparation of this article were supported by NIH Grants DC005765 to J.S.M. and HD001994 to Haskins Laboratories. We thank Jay McClelland and Jeff Elman for making the original C source code for TRACE freely available, Jay McClelland for sharing the C code and parameter files from his 1991 simulations with a stochastic version of TRACE and for helpful advice on an earlier version of this article, Jeff Elman for comments on an early version of jTRACE, and Uli Frauenfelder for sharing with us the documentation he wrote on the structure and use of TRACE. Correspondence concerning this article should be addressed to J. S. Magnuson, Department of Psychology, University of Connecticut, 406 Babbidge Road, Unit 1020, Storrs, CT 06269-1020 (e-mail: james.magnuson@uconn.edu).

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 & 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., & Hogan, E. M. (2001). Subcategorical mismatches and the time course of lexical access: Evidence for lexical competition. *Language & Cognitive Processes*, **16**, 507-534.

Elman, J. L., & McClelland, J. L. (1988). Cognitive penetration of the mechanisms of perception: Compensation for coarticulation of lexically restored phonemes. *Journal of Memory & Language*, **27**, 143-165.

Frauenfelder, U. H., & Content, A. (2000). Activation flow in models of spoken word recognition. In *Proceedings of the Workshop on Spoken Word Access Processes* (pp. 79-82). Nijmegen.

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

Frauenfelder, U. H., Segui, J., & Dijkstra, T. (1990). Lexical effects in phonemic processing: Facilitatory or inhibitory? *Journal of Experimental Psychology: Human Perception & Performance*, **16**, 77-91.

Luce, R. D. (1959). *Individual choice behavior: A theoretical analysis*. 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**, 71-91. Available at www.bcs.rochester.edu/cls/s2001v2n1/magnuson_urwpls_v2n1.pdf.

Magnuson, J. S., McMurray, B., Tanenhaus, M. K., & Aslin, R. N. (2003). Lexical effects on compensation for coarticulation: The ghost of Christmash past. *Cognitive Science*, **27**, 285-298.

Magnuson, J. S., Strauss, T. J., & Harris, H. D. (2005). Interaction in spoken word recognition models: Feedback helps. In B. G. Bara, L. Barsalou, & M. Bucciarelli (Eds.), *Proceedings of the 27th Annual Meeting of the Cognitive Science Society* (pp. 1379-1384). Mahwah, NJ: Erlbaum.

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

Marslen-Wilson, W., & Warren, P. (1994). Levels of perceptual representation and process in lexical access: Words, phonemes, and features. *Psychological Review*, **101**, 653-675.

Massaro, D. W. (1989). Testing between the TRACE model and the

fuzzy logical model of speech perception. *Cognitive Psychology*, **21**, 398-421.

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

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

McClelland, J. L., Mirman, D., & Holt, L. L. (2006). Are there interactive processes in speech perception? *Trends in Cognitive Sciences*, **10**, 363-369.

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

Miller, J. L., & Eimas, P. D. (1995). Speech perception: From signal to word. *Annual Review of Psychology*, **46**, 467-492.

Mirman, D., McClelland, J. L., & Holt, L. L. (2005). Computational and behavioral investigations of lexically induced delays in phoneme recognition. *Journal of Memory & Language*, **53**, 424-443.

Movellan, J. R., & McClelland, J. L. (2001). The Morton–Massaro law of information integration: Implications for models of perception. *Psychological Review*, **108**, 113-148.

Norris, D. (1994). Shortlist: A connectionist model of continuous speech recognition. *Cognition*, **52**, 189-234.

Norris, D., McQueen, J. M., & Cutler, A. (2000). Merging information in speech recognition: Feedback is never necessary. *Behavioral & Brain Sciences*, **23**, 299-370.

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

Samuel, A. G., & Pitt, M. A. (2003). Lexical activation (and other factors) can mediate compensation for coarticulation. *Journal of Memory & Language*, **48**, 416-434.

NOTES

1. Magnuson et al. (2001; see also Dahan, Magnuson, & Tanenhaus, 2001) and Mirman et al. (2005) will be of particular interest to readers familiar with previously reported failures of TRACE that included TRACE simulations (Marslen-Wilson & Warren, 1994, and Frauenfelder, Segui, & Dijkstra, 1990, respectively), since these studies reexamined those previous results and demonstrated that these cases were not true failures of TRACE.

2. As Frauenfelder once put it in a conference presentation (Frauenfelder & Content, 2000), the large number of parameters in TRACE are in "delicate equilibrium." Caution must be exercised when changing any parameters, since a small change in one parameter *may* result in large changes in the model's behavior, and one cannot be sure that the model will successfully perform simulations conducted with other parameter settings.

3. See Dahan, Magnuson, and Tanenhaus (2001) and Dahan, Magnuson, Tanenhaus, and Hogan (2001) for examples in which TRACE time slices were fit to real time by measuring the number of milliseconds per phoneme in stimuli presented to human subjects and relating this to the number of TRACE time slices per analogous TRACE stimulus. This crude adjustment to experiment-specific speaking rates has thus far allowed very close TRACE fits to time course data.

4. If we assume that the length of the TRACE should be roughly equivalent to echoic memory (2 sec), this requires about 33 reduplications of each word. Assuming a lexicon of 100,000 words, this translates into millions of word units and dozens of millions of connections. Although all parties, including McClelland and Elman (1986), agree that reduplication is not an ideal solution, arguments that the additional units and connections required are too great are weak, given a brain with 100 billion neurons and 100 trillion connections.

5. Primary parameters such as alpha, gamma, decay rate, and so forth could be loaded with a file, but changes to other useful parameters, such as *fslices*, which constrains the length of the maximum input string, require recompiling in cTRACE.

6. Each of the replicated figures is posted on the jTRACE Web site.

7. Note that the reported changes in word–phoneme and phoneme–word gain generate larger lexical effects than do those reported by McClelland (1991), although with the appropriate independent effects of bottom-up stimulus and context. The values required for a full quantitative replication are word-to-phoneme alpha = .015 and phoneme-to-word alpha = .03 (as specified in the code provided by J. McClelland).

(Manuscript received July 19, 2005;
revision accepted for publication October 4, 2005.)