American Psychological Association

Psychological Review

0033-295X/18/$12.00

2018, Vol. 125, No. 4, 453–485
http://dx.doi.org/10.1037/rev0000100

Automatic Control: How Experts Act Without Thinking

Gordon D. Logan
Vanderbilt University

Experts act without thinking because their skill is hierarchical. A single conscious thought automatically produces a series of lower-level actions without top-down monitoring. This article presents a theory that explains how automatic control is possible in skilled typing, where thinking of a word automatically produces a rapid series of keystrokes. The theory assumes that keystrokes are selected by a context retrieval process that matches the current context to stored contexts and retrieves the key associated with the best match. The current context is generated by the typist's own actions. It represents the goal ("type DOG") and the motor commands for the keys struck so far. Top-down control is necessary to start typing. It sets the goal in the current context, which initiates the retrieval and updating processes, which continue without top-down control until the word is finished. The theory explains phenomena of hierarchical control in skilled typing, including differential loads on higher and lower levels of processing, the importance of words, and poor explicit knowledge of key locations and finger-to-key mappings. The theory is evaluated by fitting it to error corpora from 24 skilled typists and predicting error probabilities, magnitudes, and patterns. Some of the fits are quite good. The theory has implications beyond typing. It argues that control can be automatic and shows how it is possible. The theory extends to other sequential skills, like texting or playing music. It provides new insights into mechanisms of serial order in typing, speaking, and serial recall.

Keywords: cognitive control, expertise, serial order, typing, context updating

Expertise is a paradox: Experts outperform novices but think less about what they do. How can less thinking produce better performance? The standard resolution from studies of skill acquisition and automaticity is that learning mechanisms reduce the number of things to think about (Anderson, 1982; Fitts & Posner, 1967; Logan, 1988). Chunking reduces several objects to a single representation in perception and in working memory. Memory for solutions reduces the number of steps required to solve a problem, sometimes to a single step. Learning mechanisms improve performance through some form of strengthening of associations or connections, speeding response time and increasing accuracy, so experts perform better than novices. The standard resolution accounts for a lot of data, especially on cognitive skills expressed in reaction time (RT) tasks, but it cannot account for skills in which the number of steps cannot be reduced. Skilled and novice typists must type all of the letters in each word. Skilled and novice musicians must play all the notes on the score. Skilled and novice dancers must execute the same series of steps. In these cases, experts outperform novices and do so without thinking. How can experts go through the same steps but think less? Another standard resolution is to propose hierarchical control, in which central processes do the thinking and subordinate processes do the acting (Logan & Crump, 2011; Miller, Galanter, & Pribram, 1960). Novices use central processes to control their performance, thinking through each step. Experts use subordinate processes to control their performance, reducing the number of central processing steps to one. Their central processes think less because subordinate processes take over the task of choosing and sequencing responses. However, this resolution of the paradox is incomplete. It claims *that* skilled performance is controlled hierarchically, but it does not say *what* is controlled or *how* subordinate processes control it.

The goal of this article is to provide a theory of automatic control in expert typewriting that resolves the paradox of expertise, saying what subordinate processes control (the order of keystrokes and the fingers used to strike them) and how they exert control (through context retrieval and updating). The theory explains control as memory retrieval, incorporating aspects of Logan's (1988) instance theory, Howard and Kahana's (2002) temporal context model, and Rosenbaum and colleagues' theory of motor memory (Rosenbaum, Loukopoulos, Meulenbroek, Vaughan, & Engelbrecht, 1995; Rosenbaum, Meulenbroek, Vaughan, & Jansen, 2001). The theory is a computational model implemented in a computer simulation that takes words as input and gives sequences of keystrokes as output. I evaluate the theory by comparing it with a list of desiderata and fitting it to corpora of errors generated by skilled typists to predict frequencies, magnitudes, and patterns of errors. The theory addresses typing but it would be straightforward to extend it to other sequential skills, like playing music or dancing. The theory has implications for theories of skill, automaticity,

I am grateful to Trish Van Zandt, Tom Palmeri, and Sean Polyn for help with math, Matlab, and modeling and to Jane Zbrodoff for help with the big picture. I am grateful to Jana Ulrich for testing the typists and finding what I needed when I needed it. The ideas in this paper were presented at the summer meeting of the Experimental Psychology Society in July 2017 and at the annual meeting of the Psychonomic Society in November 2017. Raw data, analysis, fitting, and simulation codes are available at https://osf.io/nj3pb/

Correspondence concerning this article should be addressed to Gordon D. Logan, Department of Psychology, Vanderbilt University, Nashville, TN 37240. E-mail: gordon.logan@vanderbilt.edu

and expertise, and for theories of serial order in language, learning, and memory.

Why Typing?

I focus on typing because it is socially relevant. Typing pervades modern culture as the primary means of interfacing with computers and the Internet. A United Nations census in 2015 found that 48% of households worldwide had computers (82% in developed countries) and 52% of households worldwide had Internet access (84% in developed countries; http://www.itu.int/en/ITU-D/Statistics/Pages/stat/devault.aspx). People spend a lot of time typing. They invest years in honing their skills and spend hours each day at the keyboard. College students typically have 10 years of typing experience and spend 4–5 hours per day at their computers (Logan & Crump, 2011).

The ubiquity of typing makes it easy to access highly skilled typists. Most college students are expert typists. Typically, college students in our samples type 70–80 words per minute (WPM; Crump & Logan, 2013; Logan & Crump, 2011), comparable with the professional typists studied in the last century (Salthouse, 1986). There is more diversity among modern typists. Many of them achieve high speeds with nonstandard typing styles (e.g., two-finger typing), which are often acquired before formal training begins (Feit, Weir, & Oulasvirta, 2016; Logan, Ulrich, & Lindsey, 2016; Rieger, 2007).

Typing is important theoretically. It is a paradigm case of sequential skills, stripped down to the basics. It challenges typists with the core computational problems that must be solved in all sequential skills: choosing the next target (key) to act on and choosing an effector (finger) to execute the action. Typing involves fewer targets (26 keys) and simpler movements (finger flexion and extension; wrist rotation) than many skills, simplifying the behavior and consequently, simplifying the chain of inference from core mechanisms to behavior. A theory of serial order and finger choice in skilled typing should generalize directly to other skills. My goal is to present such a theory at a level of abstraction that facilitates generalization.

Typing research has been guided by many theories. Some theories are qualitative, distinguishing stages (Salthouse, 1986) or levels of processing (Logan & Crump, 2011; Shaffer, 1976). Some address specific issues, like timing (Heath & Willcox, 1990; Sternberg, Knoll, & Turock, 1990; Viviani & Laissard, 1996). John (1996) and Wu and Lui (2008) presented engineering models of the human operator in typing, which assign durations to specific processes but do not explain how the processes perform their computations. In particular, they do not address serial order. The most comprehensive theory of typing is still Rumelhart and Norman's (1982) model, which proposed parallel distributed interactions among schemas for words, letters, hands, and fingers that drive the fingers to key locations on the keyboard in specific sequences. The model accounts for many error and timing phenomena. Unfortunately, important details of the model, like activation equations and connection weights, were not reported in the paper so it is not possible to test the model without reinventing it. That was part of the impetus for developing a new model. My model focuses on automatic control of serial order and finger choice. Later in the article, I consider how to extend it to account for timing (Toward a Theory of Typing) and I compare it with Rumelhart and Norman's model (Double Letter Errors in Error Patterns).

Automatic Control of Skilled Typing

Typing is hierarchical in several respects. The content we type is hierarchical. Typing expresses language, and language is structured hierarchically. Ideas are expressed in sentences, sentences are made of words, and words are made of letters. The goal structure in typing is also hierarchical: type this sentence, type these words, type these letters. The question is whether the processing structure mirrors these hierarchies. There may be many levels in the processing structure, but there is broad agreement on a hierarchical relationship between word and letter processing in skilled typists (Fendrick, 1937; Rumelhart & Norman, 1982; Salthouse, 1986; Shaffer, 1976; Sternberg et al., 1990). Logan and Crump (2011) characterize this processing hierarchy as two nested feedback loops, an *outer loop* that begins with ideas and intentions and produces a string of words to be typed, and an *inner loop* that begins with a word to be typed and produces a sequence of keystrokes. The outer loop is a central process in Fodor's (1983) sense, taking many inputs and serving many purposes. The inner loop is a module, taking only words or letters as input, giving only keystrokes as output, and serving the one purpose of producing a correct series of keystrokes.

I present a theory of automatic control of typing that specifies the memory representations and the control processes in the inner loop. It assumes the inner loop retrieves the locations of the keys to be struck and the fingers used to strike each key, and it assumes the retrieval process is instigated by the intention to type a word and proceeds automatically without top-down control until the word is typed. Evidence supporting these assumptions is reviewed below.

Words Control Expert Typing

We all begin as hunt-and-peck typists, consciously breaking words into letters, searching for each letter on the keyboard, and choosing a finger to strike it with. These processes occupy working memory while the motor system deals with one keystroke at a time (see Figure 1). Once we acquire expertise, we can type by thinking of a word and letting the inner loop do the rest. The division of labor shifts from working memory to the motor system, reducing the working memory load to a single word to be typed while increasing the load on the motor system from one keystroke to several (see Figure 1). Thus, expert typists think less but do more.

Evidence for this workload shift comes from studies that compare typing words and nonwords. Nonwords push skilled typists back on the learning curve by removing their ability to use a single chunk to type several letters. Thus, words represent skilled typing and nonwords represent novice typing. Many studies have found that words are typed much faster than nonwords (Salthouse, 1986; Yamaguchi & Logan, 2014b, 2016). Structures larger than words have little impact. Scrambling the order of words in a sentence does not slow typing speed, but scrambling the letters in a word has dramatic effects (Fendrick, 1937; Hershman & Hillix, 1965). Nonwords impose a larger working memory load than words (Yamaguchi & Logan, 2014b) and activate fewer characters in parallel in the motor system (Behmer & Crump, 2017; Crump &

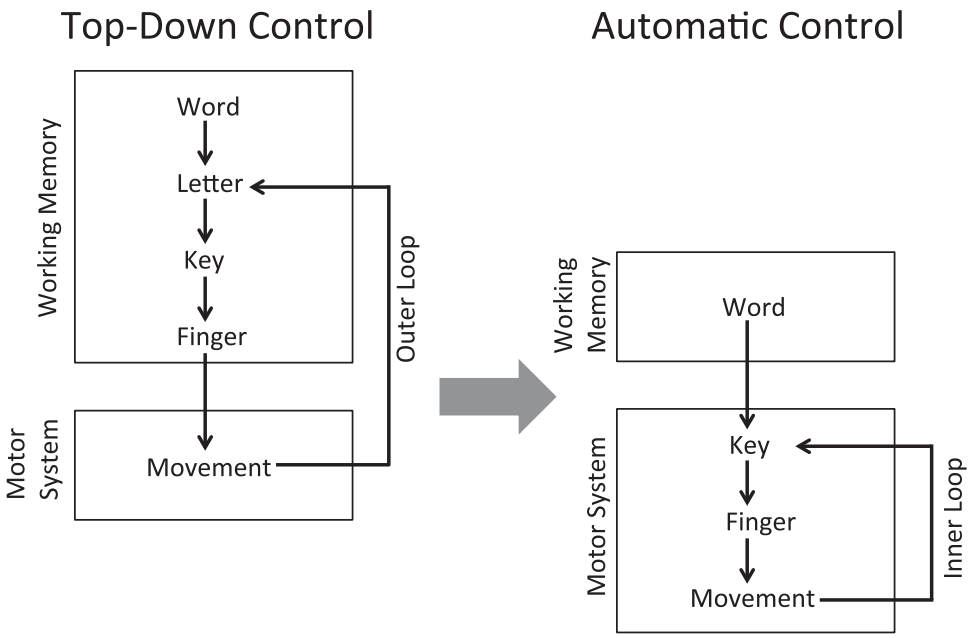

Figure 1. Typing under top-down control (left) and automatic control (right). Top-down control occurs primarily in working memory, where words are broken down into letters, the location of each letter on the keyboard is identified, and a finger to strike the key with is chosen. Top-down control places high demands on working memory and low demands on the motor system. Automatic control occurs primarily in the motor system, instigated by the presence of a word to be typed in working memory. Key and finger selection are done in the motor system. Automatic control places low demands on working memory and high demands on the motor system.

Logan, 2010b; also see Logan, 2003; Logan, Miller, & Strayer, 2011; Rieger, 2007; Rieger & Rieger, 2004).

Typing Skill Is Specific

Typing skill is largely specific to the words we know. This is evident in the common finding that skilled typists type words faster than nonwords. General skills should apply to all letter strings but typists express their skill only with familiar ones. Our experiments on learning always show an advantage of practiced over novel strings, for both words and nonwords (Crump & Logan, 2010a; Yamaguchi & Logan, 2014a, 2016). Sometimes there are advantages for repeating constituents of practiced strings, such as bigrams, but there is always an additional advantage for repeating the whole string.

The specificity is important practically because it lets us manipulate the level of automaticity within a skilled typist. It is important theoretically because it suggests that typing relies on a large vocabulary of learned sequences. A model of skilled typing must have the capacity to represent many sequences and it must specify the mechanism by which they are learned.

Experts Know Little About Keys and the Fingers That Strike Them

Skilled typists have poor explicit knowledge of the locations of the keys on the keyboard. Their explicit recall and recognition of absolute and relative key locations is much less accurate than their typing (Liu, Crump, & Logan, 2010; Snyder, Ashitaka, Shimada, Ulrich, & Logan, 2014). Their knowledge of key locations must be implicit in the motor system.

Skilled typists have poor explicit knowledge of which finger or hand types which letter. Logan and Crump (2009) had skilled typists type paragraphs striking only the keys they would strike with their right (or left) hand and omitting the others. Their typing slowed from 80 WPM using both hands to 14 WPM using one hand, and their error rate increased from 6% to 30% (also see Snyder & Logan, 2013). Knowledge of the mapping of fingers to keys must be implicit in the motor system.

Desiderata

These results suggest a list of desiderata for a model of skilled typing. Skilled typing must be instigated by a single word in the outer loop. Skilled typists must have large vocabularies of words they can type. The inner loop must know key locations and finger-to-key mappings because the outer loop does not. The inner loop must be able to retrieve keystrokes in correct serial order with no top-down intervention. The model I present fulfills these desiderata.

A Model of Automatic Control

I propose a theory of *C*ontext *R*etrieval and *U*pdating in skilled typing (CRU) that provides a computational model of automatic control in the inner loop. CRU takes words from the outer loop as input and produces sequences of keystrokes as output. CRU places little constraint on outer loop processing. The outer loop must

provide words to type, which can come from any source. Given a word, CRU (1) instigates a context retrieval process that retrieves a key location, (2) instigates a finger retrieval process that retrieves the finger associated with the location, and (3) launches a movement that strikes the key with that finger. The motor command issued in (2) is used to update the context (1), which retrieves another key and finger (2), which launches a movement (3) and updates the context again. Stages (1), (2), and (3) iterate until a space bar response is retrieved, which signals the outer loop to provide a new word to type. Outer loop control is exerted before the first keystroke and after the last one. Typing is automatic in that no top-down control is required to choose and sequence keystrokes (Tzelgov, 1997, 1999). Typing is driven by the contexts it creates with its own actions.

Many approaches to typing distinguish between stages that select and execute keystrokes (Salthouse, 1986). The distinction between selecting keys and selecting fingers is less common. I believe it is justified by the ontology of typing. We all begin by hunting for keys and pecking at them, and this goal structure may translate directly to a processing structure. It is also justified by recent studies of nonstandard typists, who often use more than one finger to strike the same key (Feit et al., 2016; Logan et al., 2016). For them, choice of key does not determine choice of finger. Typing errors also support the distinction between choosing keys and fingers. In the corpora of errors I analyze later, 56% of the letters that were struck erroneously were other letters from the word to be typed, suggesting a context retrieval error, and 23% of the errors were struck adjacent to the correct key, suggesting a finger retrieval error (also see F. A. Logan, 1999; MacNeilage, 1964).

Context Retrieval and Updating

CRU selects key locations with a context retrieval process that compares the current context to a large set of stored contexts and retrieves the letter associated with the one that is most similar (see Figure 2). The context is updated after each retrieval by adding a representation of the chosen letter to the current context (Howard & Kahana, 2002). The updated context matches a different stored context and so retrieves a different letter. This process iterates until a space bar response is retrieved, whereupon CRU sends a signal to the outer loop indicating the word has been typed and waits for the next word.

Representing words, word commands, and letter commands. Contexts are represented as vectors that are weighted sums of vectors representing word commands and letter commands. Each word and letter command is represented as a single vector with 1032 elements. All elements are set to zero except the one that represents the word or letter, which is set to 1. Thus, the length of each word and letter command vector (sum of squared element values) is 1. CRU assumes sparse, localist representations for word and letter commands, in that each one is represented by a single element in a 1032 element vector. There are 26 different letter command vectors plus a vector for the space bar, and up to 1000 different word command vectors, though I use many fewer in simulating and fitting the model.

CRU receives word command vectors from the outer loop, which initiate retrieval. CRU receives letter command vectors as output from the retrieval process. Letter commands are passed to the finger selection process to generate keystrokes and copies of letter commands are used to update the current context. Thus, the current context represents the word command and the letter commands issued so far. Context updating is like efference copy (Von Holst & Mittelstaedt, 1950) and forward modeling (Wolpert & Flanagan, 2001) in that copies of motor commands are used in computations that determine behavioral choices.

CRU assumes that words are represented as collections of context vectors that share a common word command. There are no associations between the context vectors that represent a word and no higher-order representations that link, bind, or chunk them together. They are linked by similarity instead of association. Stored context vectors are accessed in parallel, and the ease of access depends on similarity, not association.

Current context and context updating. CRU chooses letter commands by matching a representation of the current context to all stored representations of past contexts in parallel (Logan, 1988). The letter command associated with the best-matching context is retrieved, sent to the key selection process, and added to the current context using an updating rule from Howard and Kahana (2002):

$$\boldsymbol{c}_{\text{new}} = \beta \cdot \boldsymbol{r} + \rho \cdot \boldsymbol{c}_{\text{old}} \tag{1}$$

where $\boldsymbol{c}_{\text{new}}$ is the updated context vector, $\boldsymbol{r}$ is the retrieved letter command vector, $\boldsymbol{c}_{\text{old}}$ is the previous context vector, β is the weight given to new information, and

$$\rho = \sqrt{(1 + \beta^2[(\boldsymbol{r} \cdot \boldsymbol{c}_{\text{old}})^2 - 1]) - \beta(\boldsymbol{r} \cdot \boldsymbol{c}_{\text{old}})}$$

is the weight given to old information. The value of ρ is chosen to normalize $\boldsymbol{c}_{\text{new}}$ so its length equals 1. This normalization is essential in modeling serial order. The $\boldsymbol{r} \cdot \boldsymbol{c}_{\text{old}}$ term is the dot product of the letter command vector and the context vector, which represents the correlation between them. If a new letter command is added to the current context, it shares no elements with the context, so the dot product is 0 and $\rho = \sqrt{1-\beta^2}$

Figure 2 illustrates the evolution of the current context in the course of typing the word 'DOG.' Panel A illustrates it symbolically to express the basic ideas. Panel B illustrates it numerically, to show the calculations from the model. Typing begins when the outer loop sends a representation of 'DOG' to be typed. CRU has the intention to type 'DOG' but has not typed anything yet. The current context $\boldsymbol{c}$ is initialized with the vector representing 'DOG' in which one word command element is set to 1 and all of the letter command elements are set to 0. CRU compares the current context with all the stored contexts in memory in parallel, and retrieves a vector representing the letter 'D.' This vector is a letter command that is sent to the finger selection process and copied back into the current context through Equation 1. The letter command vector is multiplied by β, the current context vector is multiplied by ρ, and the two are added together. The result is an updated current context in which CRU intends to type 'DOG' and has typed 'D." The new current context matches a different stored context and retrieves 'O,' which is sent to finger selection and used to update the current context. The updated context retrieves 'G,' which changes the context again, so the space bar response is retrieved and CRU tells the outer loop it finished the word.

Serial order is controlled by the evolution of the current context and the changing similarities to stored contexts. CRU is driven by

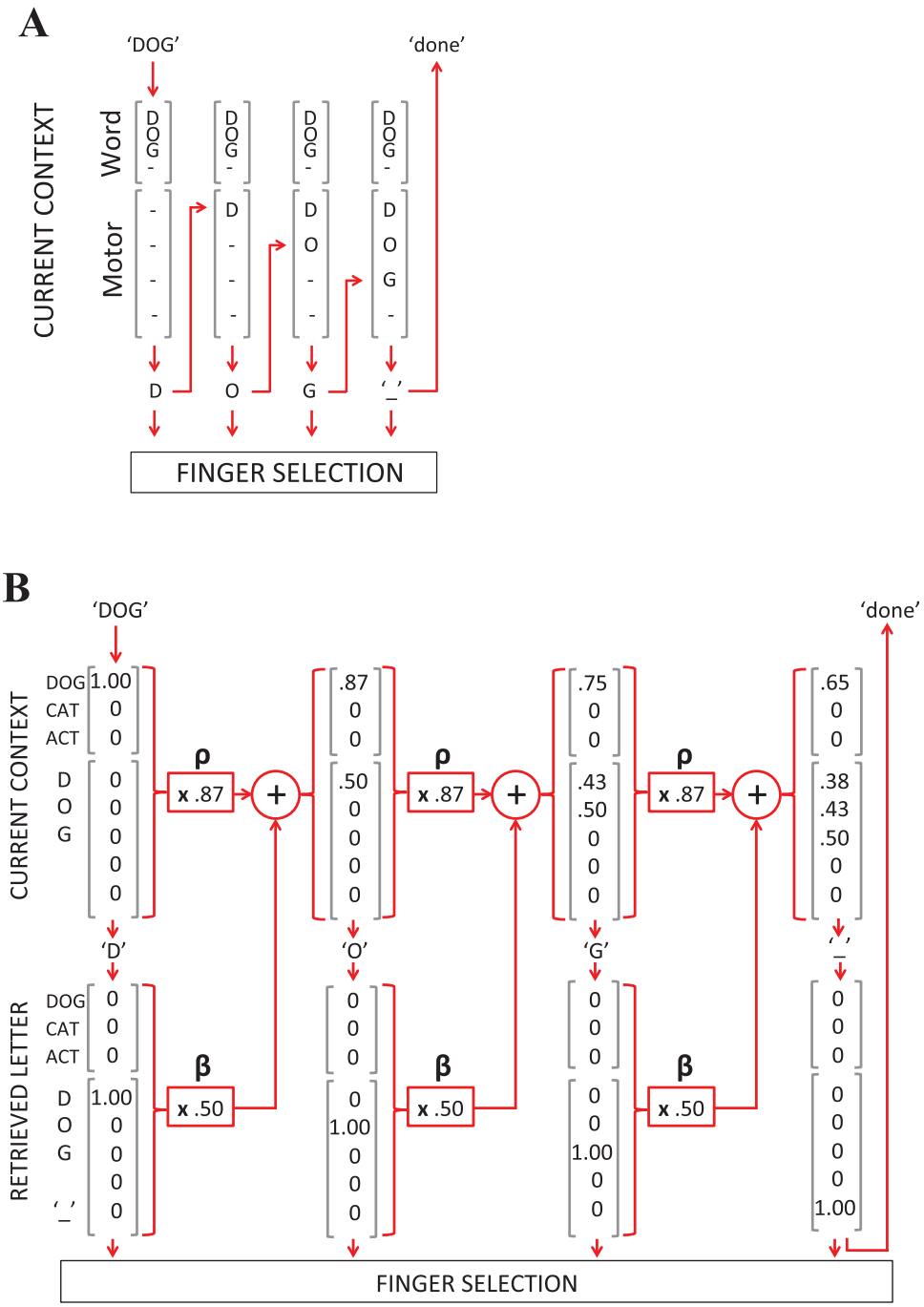

Figure 2. Evolution of the current context over time while typing 'DOG.' Red arrows represent the sequence of events. Panel A represents the context vectors symbolically to facilitate understanding. Panel B represents the context vectors numerically, as in the model. Top-down input from the outer loop ('DOG') sets the current context to the word to be typed (left vector in A; top left vector in B). The current context retrieves 'D' (symbol in A; bottom left vector in B), which is sent to the finger selection process and copied back into the current context. It is multiplied by β and added to ρ times the current context, producing an updated current context vector (second vector in A; second from left in top row in B). The new context retrieves a different letter. This process repeats until the space bar response is retrieved, which is sent to the outer loop to indicate that typing is done.

the contexts it generates with its own behavior. Many models of serial order assume retrieval is driven by context. CRU is different in assuming that the context is self-generated (also see Howard & Kahana, 2002; Lohnas, Polyn, & Kahana, 2015; Polyn, Norman, & Kahana, 2009).

Stored contexts, similarity, and context retrieval. CRU assumes that the current context vectors are associated with the letter commands they retrieve and are stored in long-term memory in the motor system. The idea that responses are associated with the contexts in which they are chosen is common in theories of skill acquisition (e.g., Logan, 1988). CRU's context retrieval process generates a potential set of instances—a set of current contexts—each time it types a word. Thus, skilled typists have a large number of stored contexts representing a large vocabulary of words acquired over 10 or more years of experience (Logan & Crump, 2011). There is a memory representation for each context vector involved in typing a word, so there are $N + 1$ context vectors for an N-letter word.

The stored contexts and associated letter commands for typing 'DOG' are illustrated symbolically in the top panel of Figure 3 and numerically (as in the model) in the top panel of Figure 4. The ordering in the figures is only for graphical convenience. The memory representations are stored and retrieved independently, and retrieval is parallel. Stored context vectors are not associated with each other and letter commands are not associated with each other. The only associations are between context vectors and letter commands (the arrows in the figures), which represent CRU's knowledge of key locations in the inner loop (Liu, Crump, & Logan, 2010; Snyder et al., 2014). Thus, the sequence of CRU's choices depends more on similarity than association.

The context retrieval process is depicted in the middle panels of Figures 3 and 4. These examples represent beginning to type the word DOG. The current context contains the word to be typed and no copies of letter commands. The current context is matched to each of the stored contexts independently and in parallel. Similarity drives retrieval. Stored contexts compete for retrieval in proportion to their similarity to the current context. With sparse representations like CRU's, similarity is determined approximately by feature overlap. Two vectors are compared, and a point is taken away for each mismatching feature. This can be seen in the middle panel of Figure 3, in which the current context is compared with each of the stored contexts for typing 'DOG.' The model uses the *dot product* between vectors as a measure of similarity. Formally, the dot product between two vectors $\boldsymbol{c}_{\mathrm{A}}$ and $\boldsymbol{c}_{\mathrm{B}}$ is the sum of the products of corresponding elements:

$$\text{dot product} = \sum_{i}^{N} \boldsymbol{c}_A(i) \times \boldsymbol{c}_B(i).$$

The middle panel of Figure 4 illustrates how the dot product is calculated. The bottom panels of Figures 3 and 4 show the values of the dot product between the current context representing the intention to type DOG without having typed anything and the stored contexts for typing DOG. Similarity is high for the stored context representing the first letter of the word and declines progressively for subsequent letters.

Figure 5 shows the similarities between all the stored contexts for typing DOGS, coded by color. For example, the blue line in the graph represents the similarity of the middle context (intend to type DOGS; D and O typed) to all other contexts. It shows a symmetric gradient of similarity that decreases with distance from the middle context. The steepness of the gradient depends on the value of β in the updating equation (Equation 1). It is steeper for higher values of β, which represent greater emphasis on the new keystroke and less emphasis on past keystrokes (see Figure 6).

Figure 5 illustrates the competition for retrieval in each context. Imagine that the blue context is the current context that is matched to all the stored contexts. The blue line shows the dot products between the blue context and all the others, which represent strength with which each context competes for retrieval. The blue context is the strongest competitor, but green and magenta contexts also compete strongly, as they are quite similar to the blue context. The red context competes less strongly, as it is less similar to the blue context.

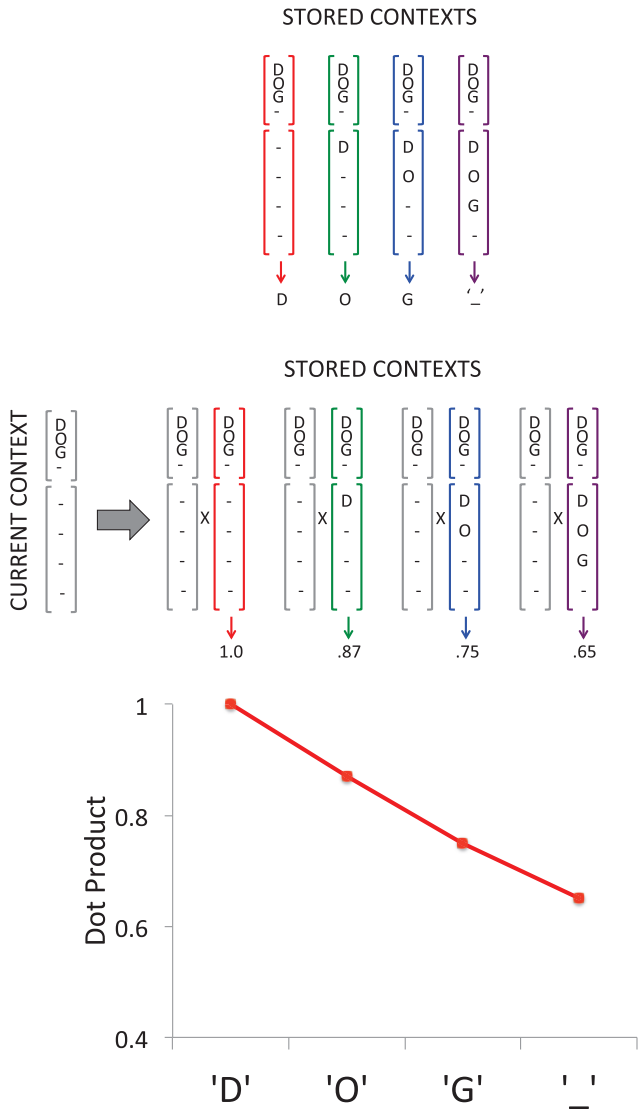

Figure 3. Symbolic representation of stored contexts (top), matching the current context to the stored contexts (middle), and the similarity of the current context to each stored context (bottom).

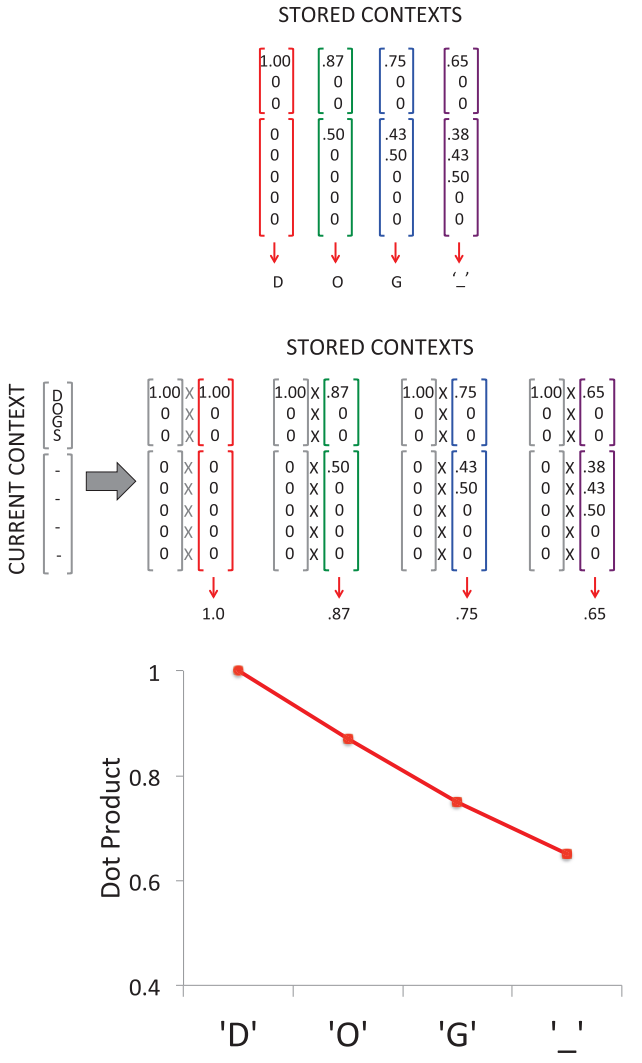

Figure 4. Numerical representation (used in the model) of stored contexts (top), matching the current context to the stored contexts (middle), and the similarity of the current context to each stored context (bottom).

Choice. The choice process is modeled as a race between N stochastic accumulators, one for each stored context in long-term memory (Logan, Van Zandt, Verbruggen, & Wagenmakers, 2014). Each runner is modeled as a diffusion to a single threshold. Its finishing time distribution is the Wald, which has two parameters: drift rate v and threshold θ. Its probability density function is

$$f(t) = \theta(2\pi t^3)^{-1/2} exp\left[-\frac{1}{2t}(vt-\theta)^2\right]. \quad (2)$$

The runners race independently, so the finishing time distribution for the winner i is

$$f(t,i) = f_i(t)\prod_{j\neq i}^{N}[1-F_j(t)] \quad (3)$$

and the probability of i winning is

$$P(i) = \int_0^\infty f(t,i)\,dt. \quad (4)$$

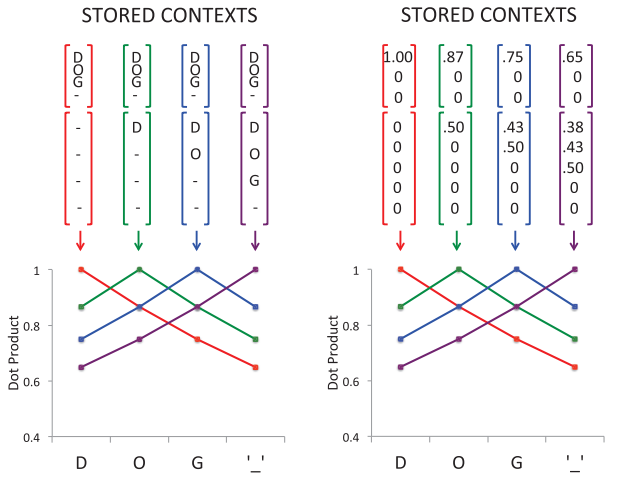

Figure 5. Similarities of each stored context to every other stored context (symbolically on the left; numerically on the right). Colors in the graph correspond to colors in the stored contexts. The blue line represents the similarity of the blue context (intend to type DOGS, have typed D and O) to all other contexts. Similarity is graded around the best-matching context.

The finishing time distribution for a race between diffusions is given by substituting the Wald distribution for the generic distributions in Equation 3:

$$f(t,i) = \theta_i(2\pi t^3)^{-1/2}\exp\left[-\frac{1}{2t}(v_i t-\theta_i)\right]\times\prod_{j\neq i}^{N}\left[1-\Phi(t^{-1/2}(v_j t-\theta_j))\right.$$
$$\left.-\exp(2v_j\theta_j)\Phi(-t^{-1/2}(v_j t-\theta_j))\right] \quad (5)$$

where Φ is the cumulative normal distribution. Equation 5 is a likelihood function, which I used to fit the model to empirical data. The model predicts speed and accuracy, like contemporary models of response time (Ratcliff & Smith, 2004; Teodorescu & Usher, 2013). I used it to estimate the likelihood of correct and error keystrokes in a corpus of typing errors.

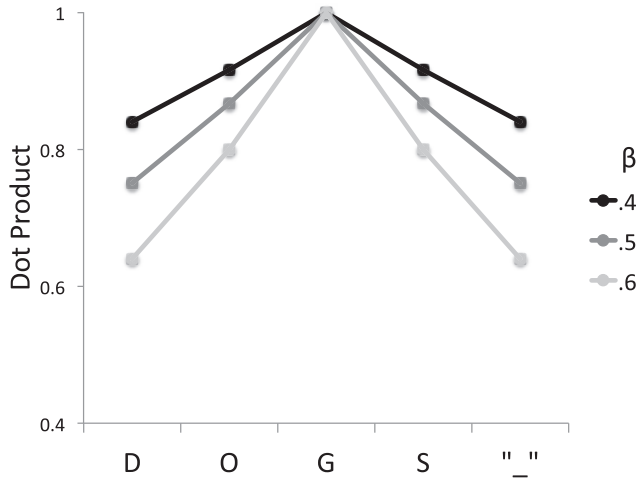

Figure 6. The effect of weighting the present (manipulating β) on the similarity gradient in the context retrieval process.

The drift rates v in Equation 5 are not free parameters. They are determined by the similarities between the current context and the N stored contexts, represented as the dot products between current and stored context vectors (see Figure 5). The context updating parameter β affects the steepness of the gradient around the best-matching context but it does not change the ordering of the similarities (see Figure 6). The similarities exert strong constraints on the model's predictions.

Word and letter contexts. Word and letter contexts play different roles in the model. The word context lets CRU focus on the stored contexts it needs to type the word, and the letter context determines the order in which CRU retrieves them. This is illustrated in Figure 7, which depicts stored contexts for typing MEAL, MEAT, REAL, and LAME (symbolically in the top panel; numerically in the middle panel) and the dot products between the contexts for typing MEAL and all the other contexts (bottom panel), which represent the competition for retrieval.

Figure 7 shows how the word command selects appropriate contexts. The first three letter commands for MEAL and MEAT

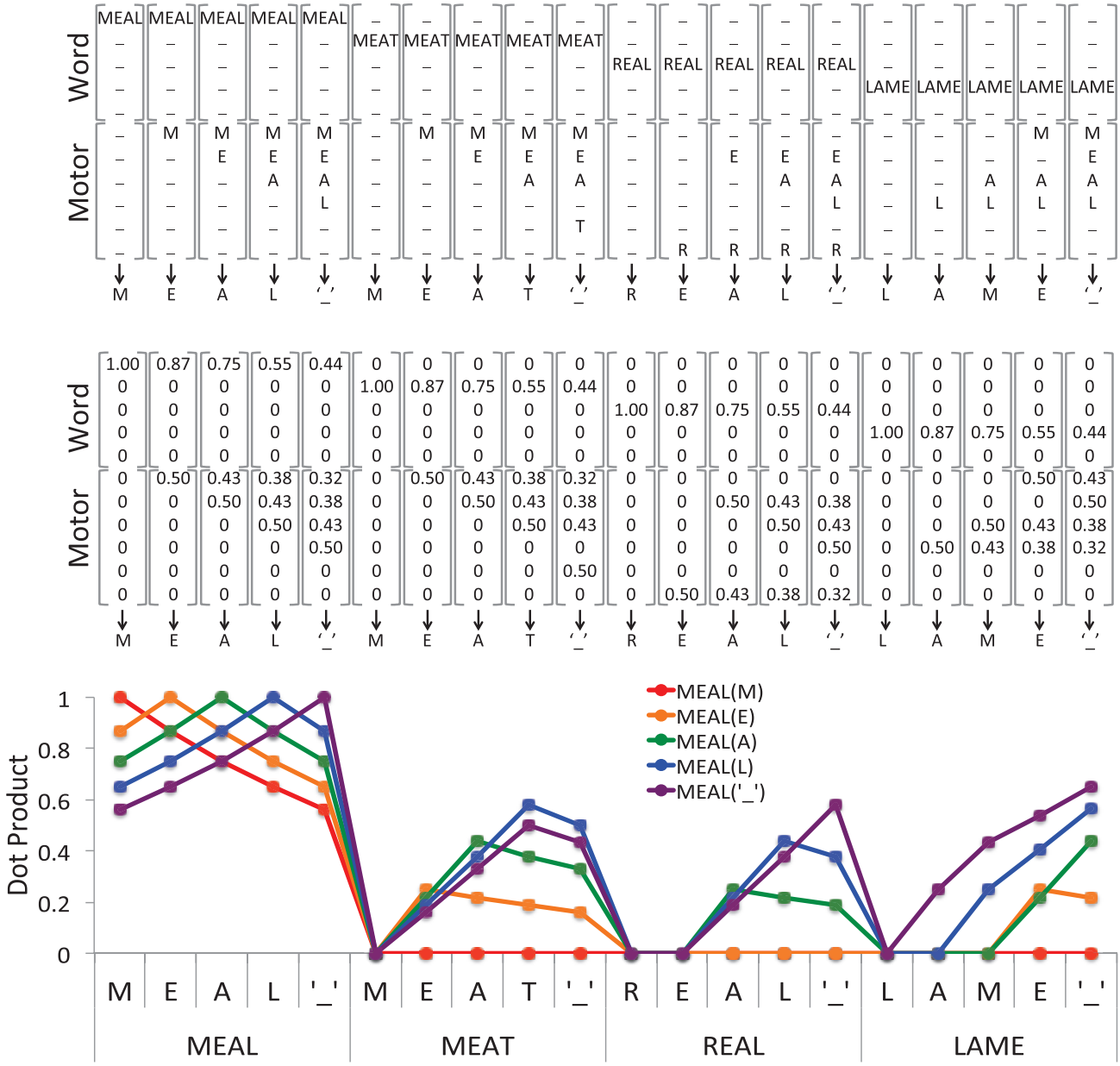

Figure 7. Stored context vectors for typing MEAL, MEAT, REAL, and LAME (symbolic in the top panel and numeric in the middle panel) and dot products between the stored context vectors for MEAL and all the stored contexts in memory (bottom panel). Dot products are higher for contexts that represent MEAL than for contexts that represent other words. Dot products are higher for letters in the same order and position as MEAL (MEAT, REAL) than for letters in a different order and different positions (LAME).

are the same but the dot products are much higher for the contexts for typing the intended MEAL than for the contexts for typing the unintended MEAT. The word contexts for MEAL and MEAT are different, and that reduces the similarity between their stored context vectors. The red line shows the similarity of the first context in typing MEAL to all the other stored contexts. MEAL is represented in the word context and there are no letters in the motor context. Similarity is high for stored contexts that contain MEAL in their word contexts and zero for all other contexts, as there is no overlap in word or motor contexts. Other contexts that contain letter commands as well as the word command produce higher dot products in other words, but the dot product for the first context is always zero because the first context contains only a word command and no letter commands, and the word command is different from all other word commands.

CRU assumes the similarity between stored contexts representing different words is determined by the overlap in letter commands. The dot product between contexts representing words that contain different letters is zero, which essentially removes those contexts from the retrieval competition. When a letter is typed, its letter command becomes part of the current context and influences all subsequent retrieval attempts. Letters that appear in the same order in different words may facilitate retrieval of the intended letter. A current context containing the intention to type MEAL and the letter command for M is most similar to the context associated with E in the intended word MEAL and the unintended word MEAT. Erroneous retrieval of E from MEAT would produce the correct response for typing MEAL. The current context after typing M and E is most similar to the contexts associated with A in MEAL, MEAT, and REAL. This facilitation might be lost when the same letters appear in different orders in different words (compare MEAL and LAME). However, the word context gives a strong advantage to letters in the intended word, diminishing these effects. Exploratory simulations with the best-fitting parameters from the fits reported later show no effect of letters in unintended words.

The word context facilitates typing by adding dimensions that distinguish intended motor sequences from similar unintended ones (Logan, 2002). Unwanted sequences are not suppressed. Differences in the word context make them less similar, so they are less likely to be selected. The word context allows the model to represent large vocabularies of words and reliably select the ones that the outer loop intends, like skilled typists.

Finger Selection

Context retrieval provides the location of the key to be struck. Finger selection chooses a finger and the posture required to strike the key (Rosenbaum et al., 1995, 2001), after which a stereotyped movement is launched (Flanders & Soechting, 1992; Soechting & Flanders, 1992). I assume that the retrieved location serves as a cue to retrieve the finger and posture. I assume that fingers and postures are associated with locations in accord with frequency of use. Standard typists will have one finger associated with each location. Nonstandard typists may have two or more (Feit et al., 2016; Logan et al., 2016). Fingers and postures can be selected by other means, such as visual guidance in hunt-and-peck typing, but I assume skilled typists rely on memory retrieval. The memory traces that support retrieval represent CRU's knowledge of which fingers strike which keys in the inner loop (Logan & Crump, 2009; Snyder & Logan, 2013).

Finger selection is implemented as a race between $26 \times M$ diffusions following Equations 2–5, where M is the number of fingers a typist uses. $M = 8$ for standard typists, $M < 8$ for nonstandard typists, and $M = 2$ (thumbs) for texting on smart phones. The drift rate in each diffusion depends on the strength of association A_{ij} between the finger and the location and the strength of activation of the location L_i. I assume that location is represented as a spatial gradient that peaks at the center of a key and decreases exponentially with distance (Logan, 1999; Shepard, 1987; see Figure 7). The strength of activation L_i for location i is

$$L_i = \exp(-S \cdot d_{ij}) \tag{6}$$

where S is a sensitivity parameter and d_{ij} is the Euclidean distance from the center of location i and the center of the retrieved location j provided by context retrieval. Drift rate v_{ik} for finger k at location i is the product of association strength A_{ik} and location activation:

$$v_{ik} = L_i A_{ik}. \tag{7}$$

The model assumes equal association strengths between standardly mapped fingers and keys in skilled typists, so drift rate depends only on distance and sensitivity. Figure 8 shows a gradient of activation along the home row when the target key is G. The steepness of the gradient depends on the sensitivity parameter S.

The gradient allows the model to make errors adjacent to the intended key, which occur frequently in typing. Interestingly, CRU predicts that these errors will be struck with the finger appropriate to the key. Adjacent errors are not mis-aimed movements. They are mis-selected finger postures. Video analyses of skilled typists show that adjacent errors are often made with an appropriate finger, as CRU predicts (Grudin, 1983).

Evaluating the Model

CRU fulfills the desiderata for a model of automatic control: Typing is instigated by a single word from the outer loop, which sets the word command in the current context and begins matching the current context to stored contexts. The word command provides a common thread that links the required letter commands through similarity and allows the model to type large vocabularies of words. The context retrieval process knows key locations and the finger selection process knows finger-to-key mappings. The context updating process and the similarities among stored contexts allow the model to retrieve keystrokes in correct serial order without top-down control. The spatial gradient in retrieved locations allows the model to make errors on keys adjacent to intended ones.

CRU is implemented as a computer simulation that takes arbitrary word lists or texts as input and gives sequences of keystrokes for each word as output. The simulation begins by training the model on a corpus of words, establishing a set of stored contexts for each of the words. This involves running each word through the context updating process to generate a set of contexts to store and associating the contexts with the corresponding letter commands. CRU does not model the learning process, though learning is fertile ground for theoretical development. For convenience, CRU adopts the heuristic assumption that learning is complete after one pass through the corpus.

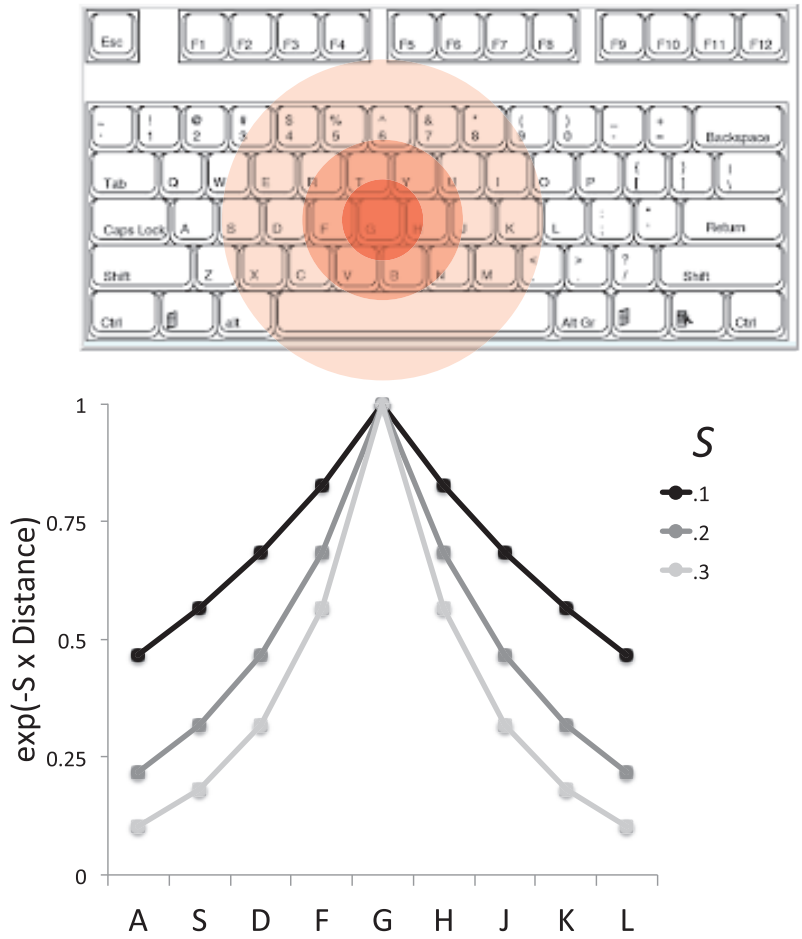

Figure 8. Representation of locations in the finger choice process peaks at the center of the intended key and decreases exponentially with Euclidean distance from that key ($L_i = \exp[-S \cdot d_{ij}]$). The graph shows the effect of the sensitivity parameter S on the steepness of the gradient around G across the home row of the keyboard.

The simulation types each word by forming an initial context that represents the intention to type a word, instigating context retrieval, choosing a letter command and using it to select a finger and update the current context. The process iterates on its own with no top-down monitoring until it retrieves the space bar command, which signals the outer loop that typing is finished. With appropriate parameters, the simulation types with high accuracy without top-down control. Thus, CRU demonstrates automatic control (Tzelgov, 1997, 1999). The question is whether it does so in the same manner as skilled typists.

I evaluated CRU by fitting it to error corpora and testing its predictions about error frequencies, magnitudes, and patterns. I used Equations 2–7 to estimate the likelihood of each keystroke in a corpus and maximized the likelihood for each corpus. CRU gives higher likelihoods for correct responses than for errors, but it gives higher likelihoods for some errors than for others (e.g., adjacent vs. nonadjacent transpositions), so CRU will fit better when typists make errors it predicts. After the fitting, I simulated the model for each typist using their best-fitting parameters to generate a set of predicted errors and I calculated summary statistics for the predicted and observed errors for each typist. To test predictions about serial order, I calculated transposition gradients to show the tendency to type keystrokes in nearby positions and lag conditional recall probability functions to show the tendency to type successive letters consecutively. To compare the magnitudes of errors and assess recovery from errors, I calculated Levenshtein (1966) and Damerau (1964) distances between correct and error strings for typists and the model. To compare the types of errors, I categorized human and model errors using model-based and traditional taxonomies. I compared CRU's account of doubled letter errors (MMET for MEET) with Rumelhart and Norman's (1982).

CRU predicts the accuracy and latency of individual keystrokes. The summary statistics reflect emergent properties of the model that depend on the similarity structures in the stored contexts and the spatial constraints of the keyboard. The summary statistics are parameter-free predictions of the model in that they require no further parameter estimation or adjustment. The model was fit to individual keystrokes, not the summary statistics. The best-fitting parameters were not adjusted to fit the summary statistics. The best-fitting parameters drove a simulation that produced sequences of correct and error keystrokes, and the summary statistics were calculations on the simulated sequences with no further parameter

estimation. Thus, all the predictions for each typist are based on the same single set of parameters. The same calculations were performed on each typist's sequences of correct and error keystrokes to provide the observed summary statistics.

I focused the model evaluation on accuracy because errors are diagnostic of the control system, and my main goal is to test CRU as a model of automatic control. CRU predicts error frequencies, magnitudes, and patterns, and those predictions must be tested. These summary statistics provide critical tests of theories of serial order (e.g., Dell, 1986; Dell, Burger, & Svec, 1997; Hurlstone, Hitch, & Baddeley, 2014; Lewandowsky & Farrell, 2008; MacNeilage, 1964; Rumelhart & Norman, 1982). CRU predicts latency as well as probability. I report latency results and model predictions later (see Toward a Theory of Typing and Keystroke Timing below).

The Error Corpora

To generate sets of errors for analysis, I had 24 skilled typists type one of four short paragraphs on the many merits of border collies 20 times. The paragraphs were 106, 113, 107, and 108 words long. The mean WPM and error rate across typists are plotted for each repetition in Figure 9. Mean WPM, error rate, and other measures for each typist are presented in Table 1. The details of the method and the paragraphs can be found in Appendix A. Mean typing speed was 81.9 WPM. It increased slightly over repetitions, from 79 to 85 WPM, $F(19, 437) = 3.14$, $MSE = 15.25$, $p < .01$, which amounts to an 11-ms reduction in interkeystroke interval. Mean error rate was 7.3% and was relatively stable over repetitions, $F(19, 437) = 1.07$, $MSE = 6.76$, $p = .38$. Assuming independent keystrokes and 5 letters per word, this corresponds to an error rate of 1.5% for each keystroke (i.e., $P_W = P_K^5$ so $P_K = P_W^{(1/5)}$ and error rate $= 1 - P_K$). Skilled typists retrieve keystrokes very accurately.

On average, typists typed 117 words erroneously over the 20 presentations. I reduced the set of error words for each typist by (a) including only 3–8 letter words, which were most frequent in the texts and in the language, (b) excluding repetitions of the same error to focus on unique errors, and (c) including only words with single errors and transpositions (77% of the errors in the corpora) because the source of a single error is easy to identify but the sources of multiple errors are not. This reduction resulted in a set of words with unique errors for each typist (see Table 1) with an average of 67.8 unique error words (≈ 339 keystrokes) per typist. I fitted CRU to the unique error words for each typist.

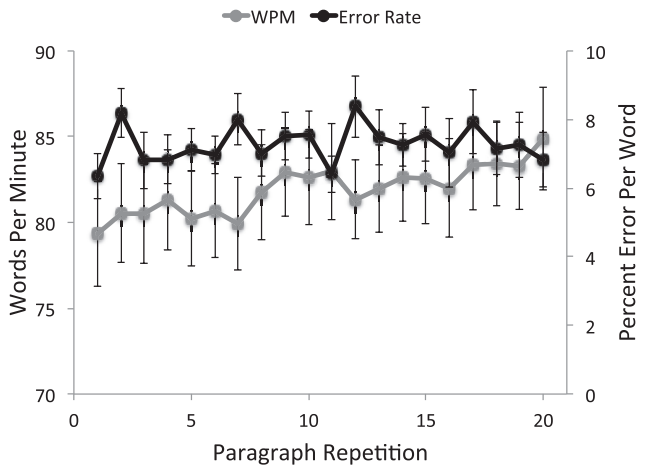

Figure 9. Words per minute and percentage of error per word for the 20 repetitions of paragraph typing averaged over 24 skilled typists. Error bars are standard errors of the mean.

Fitting the Model

I fit CRU to each typist's corpus of unique error words with maximum likelihood estimation. I fit individual keystrokes, calculating the likelihood of context retrieval and finger retrieval for each keystroke in each erroneously typed word (Kragel, Morton, & Polyn, 2015; Morton & Polyn, 2016). To do so, the fitting routine used Equation 1 to build a set of context vectors for typing the word correctly with β as a free parameter. Then it stepped through the erroneous word, building the current context with Equation 1 using the same value of β.

First, the fitting routine calculated the likelihood of retrieving the correct or erroneous letter, P_L. It computed the dot product between the current context in the erroneous word and each stored context for that word to get drift rates, and used Equations 4 and 5 to compute the likelihood for the key that was struck. Letters in the correct position got the highest likelihood. Letters (errors) from within the word got lower likelihood (see Figure 5), and letters from outside the word got zero likelihood.

Second, the fitting routine calculated the likelihood of retrieving the finger, P_F, given the letter command from context retrieval. I assumed the finger was chosen correctly ($P_F = 1$) if the context retrieval chose a wrong letter from the same the word. If the error was not from the word, I assumed context retrieval chose the right letter ($P_L = 1$) but letter retrieval failed ($P_F < 1$). In this case, the fitting routine calculated the distance between the correct letter and each letter on the keyboard to generate drift rates with Equation 6, and then calculated the likelihood of the erroneous letter given the drift rates using Equations 4 and 5. I excluded locations more than 4 cm from the correct location (3–4 key widths) because they likely came from other sources, like misaligning the fingers on the keyboard, and so were outside the scope of the model. Within the 4-cm window, the model produces higher likelihoods for locations nearer to the correct one (see Figure 8).

The model assumes context retrieval and finger selection are stochastically independent sequential stages, so the probability of typing a given keystroke P_K is the product of the probabilities of the stages: $P_K = P_L \cdot P_F$. The model assumes that successive keystrokes are chosen independently, so the probability of typing a sequence of keystrokes is the product of the probabilities of typing each of the constituent keystrokes. For an N letter word, $P_W = P_{K1} \cdot P_{K2} \ldots P_{KN}$. This probability represents the likelihood of the sequence of correct and erroneous keystrokes in the erroneously typed word. The likelihood of typing all the erroneous words is the product of these likelihoods for each word. For M erroneous words, $P_{SEQUENCE} = P_{W1} \cdot P_{W2} \ldots P_{WM}$. The likelihood for erroneous words does not reflect the occasions on which typists typed the words correctly. To compensate for this I calculated the likelihood for typing the error word correctly and multiplied it by a number representing the ratio of correct to error words (see Appendix B). I multiplied the likelihood for the correct word

Table 1
Speed, Error Rate, Total and Unique Errors, Best Fitting Parameters, and Goodness of Fit

Typist	Text	WPM	Percent error	Total errors	Unique errors	β	*S*	−Log likelihood
1	5	63.9	6.35	158	68	.5411	.1839	905.6
2	8	87.3	8.18	103	65	.5573	.2232	719.0
3	6	81.9	6.81	112	73	.5254	.1943	2181.0
4	7	72.1	6.84	107	63	.5445	.1818	973.9
5	7	85.9	7.13	85	53	.5509	.2082	1096.5
6	5	78.6	6.98	113	67	.542	.2001	944.8
7	6	90.1	8.01	146	87	.5435	.1878	1194.4
8	8	83.8	7.01	184	113	.5234	.1787	1631.1
9	8	72.0	7.51	91	61	.5507	.204	813.6
10	6	93.6	7.57	145	84	.5383	.2077	1183.4
11	7	84.4	6.41	201	104	.5183	.1754	1704.3
12	5	98.3	8.38	68	44	.566	.1976	592.6
13	7	92.3	7.48	91	44	.5506	.2103	597.0
14	6	76.3	7.26	88	61	.5279	.1943	2071.6
15	8	53.7	7.57	212	90	.5261	.1706	1193.1
16	5	78.9	7.04	103	56	.5442	.1897	907.7
17	7	103.1	7.93	137	70	.5423	.1844	1163.6
18	6	96.0	7.17	23	12	.6826	.238	377.0
19	8	90.1	7.25	61	40	.563	.2157	656.5
20	5	84.9	6.84	77	42	.514	.1996	1773.3
21	8	61.9	6.35	136	82	.5251	.1798	1786.3
22	5	78.9	8.18	132	72	.5214	.1821	1724.9
23	7	96.6	6.81	71	45	.5695	.1947	616.4
24	6	61.4	6.84	245	132	.5296	.1934	1461.8
Mean		81.6	7.29	117	67.8	.5457	.1956	1177.9
SD		12.8	.033	52.3	26.0	.0330	.0160	506.2

Note. WPM = words per minute.

by the likelihood for the error word. The fitting routine calculated logs of the likelihoods, summed them over stages, keystrokes, and correct and error words, and minimized the negative sum of the log likelihoods using Simplex (Matlab's fminsearch). The details of the fitting procedure are given in Appendix B. Thus, the model was fit to an average of 67.8 error words (≈ 339 keystrokes) and 4527.8 correct words per typist.

Two parameters were allowed to vary in the model fits: the β parameter in context retrieval and the *S* parameter in finger selection. These parameters modify the steepness of the temporal and spatial gradients, respectively (see Figures 6 and 8), and so were the most important theoretically. To avoid parameter tradeoffs, the thresholds for the two stages were fixed at values that produced good accuracy in exploratory simulations (500 for context retrieval; 200 for finger selection). The drift coefficient, which measures the standard deviation of within-trial noise in the diffusions, was fixed at 1.0 following convention, and so disappeared from the equations. The drift rates were not free parameters. They were determined by the similarity structure in the stored contexts and by the spatial structure of the keyboard. The β and *S* parameters modulated the steepness of the gradients but did not change the rank ordering of the drift rates. Thus, the model was strongly constrained.

I fit CRU to each of the 24 typists individually. Each fit took about 2 hours and converged after 50–60 iterations. The best fitting β and *S* parameters and the minimum negative log likelihood for each typist are presented in Table 1.

Generating Predictions

To generate predictions for each typist, I simulated the model using each typist's best-fitting β and *S* parameters. Each simulation used a list of the unique 3–8 letter words in the paragraph the typist typed (59, 65, 53, and 47 words for lists 5, 6, 7, and 8, respectively), so CRU and the typists were tested on the same words. The simulation generated a set of stored contexts that included each of the words on the list and then simulated typing each word 1000 times.

The simulation of each word began by setting the element representing the word command to 1 in the current context and setting other elements to zero. This initial current context was compared to each of the stored contexts, generating a dot product that represents the drift rate of the diffusion process for that context. The drift rates and threshold were used to generate a set of retrieval times by sampling from Wald distributions (Equation 2). The key associated with the context with the shortest retrieval time was chosen whether it was correct or erroneous, passed to the finger selection process, and copied into the current context vector. The response was represented as a vector with 1 in the position corresponding to the letter and 0 elsewhere. It was combined with the current context vector using Equation 1. The updated current context was compared with stored contexts and retrieved the next key to be struck. This process repeated without any external (top-down) control until the space bar response was retrieved, indicating the end of the word, or 20 characters had been retrieved (maximum word length = 8 characters). The program always terminated with a space bar response before typing 20 characters.

This part of the simulation reflects the model acting without thinking. It sequences keystrokes driven only by the similarity relations among current and stored contexts. Setting the word command in the initial context vector is all that is required to initiate the process.

The chosen key was then used to drive the finger selection process. The simulation calculated the distance between the chosen key and the 26 keys on the keyboard to generate activations for each location (L_i in Equation 6). The simulation assumed standard mapping, so the associative strengths A_{ik} in Equation 7 were set to 1 for the standardly mapped finger and 0 otherwise. Drift rates for each finger and location (Equation 7) were used to generate 26 samples from Wald distributions, and the finger and location with the minimum retrieval time, whether correct or erroneous, was selected as the keystroke for the current context.

On average, CRU simulated 56,000 words per typist, taking about 2 hours for each of the 24 typists. Errors occurred in 4132.4 (SD = 1394.8) words. Excluding repetitions and including only single errors and transpositions (Damerau distance = 1) reduced the number errors per typist to 793.4. These errors were used to generate predictions for summary statistics.

Parameter Recovery

I conducted a parameter recovery study to determine how accurately the fitting routine estimated the β and S parameters. I fitted the model to the predicted errors produced in the simulations described above and compared best-fitting parameters to the parameters I used to generate the predicted errors. For each simulated typist, I reduced the set of total error words to a set of unique single or transposition errors, and then I randomly sampled the same number of errors from the unique set that I fitted for that typist (Table 1 column 6). I used the same sample size in recovery as in the initial fitting to see how good parameter recovery was in this data set. Larger sample sizes would likely show better recovery (White, Servant, & Logan, 2017) but would take a long time. Fitting the simulated data with the initial sample sizes took about 1.5 hours for each of the 24 typists. Broader ranges of parameter values might also show better recovery. I restricted the ranges to those found in my sample of 24 skilled typists.

Figure 10 plots the initial and recovered values for the β and S parameters. Initial and recovered β parameters correlated 0.589, which is good but not perfect. Removing the rightmost point in the figure as an outlier (typist 18 who had only 12 unique errors) reduced the correlation to 0.576. Recovery was better for the S parameter. The correlation between initial and recovered parameters was 0.814. Given the small sample sizes, I find these results encouraging.

Overall Accuracy

The overall error rate for the simulation (7.3% ± 2.3%) was the same as the overall error rate for the typists (7.3% ± 3.3%). The correlation between observed and predicted values was 0.586 (see Figure 11). The predictions were based on variations in drift rate, produced by manipulating β and S. Typists may differ in threshold as well. Estimating threshold and drift rate may improve the correlation.

Serial Order: Transposition Gradients

Theories of serial order in serial recall tasks like digit span predict the probability with which an item from a given position in the list is reported in each possible position in the list. These

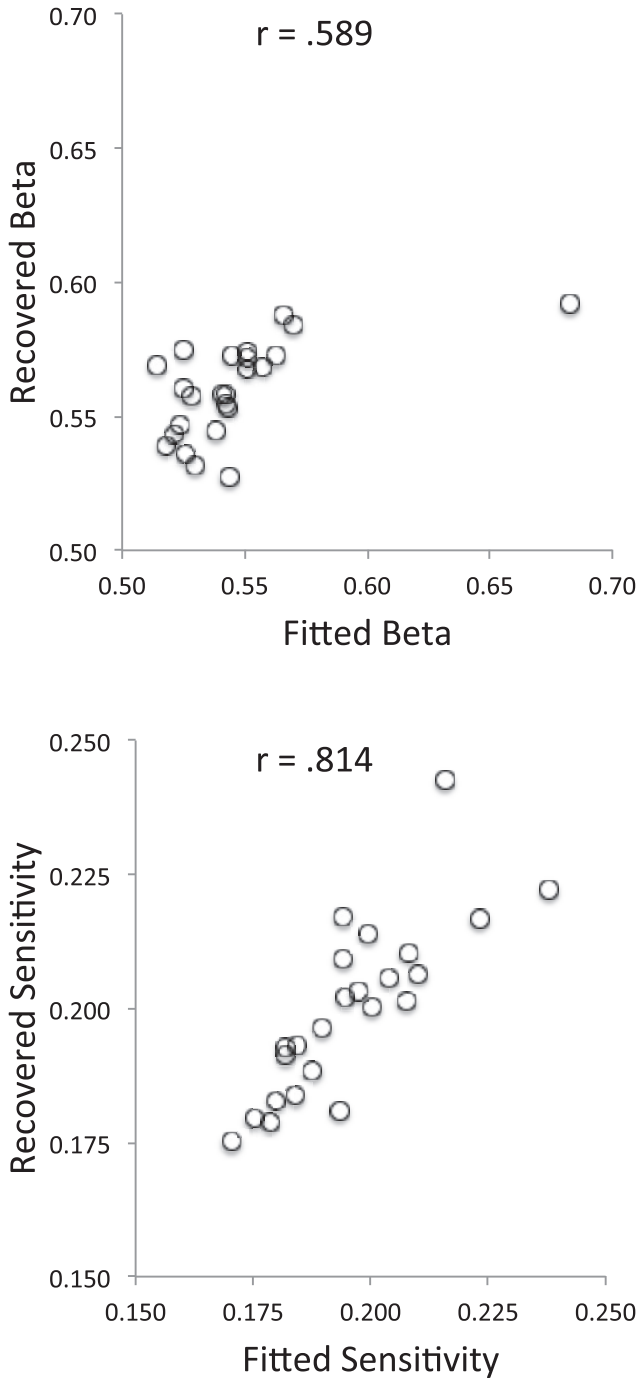

Figure 10. Fitted and recovered β (top panel) and S parameters (bottom panel). Each point represents one of 24 typists.

probabilities are often shown as *transposition gradients*, which plot the probability that each letter is recalled in each position (Estes, 1972). In empirical transposition gradients, items are more likely to be recalled in positions adjacent to the correct position than in more remote positions. This *locality constraint* is a benchmark prediction that all theories of serial recall must address (Henson, Norris, Page, & Baddeley, 1996; Lewandowsky & Farrell, 2008).

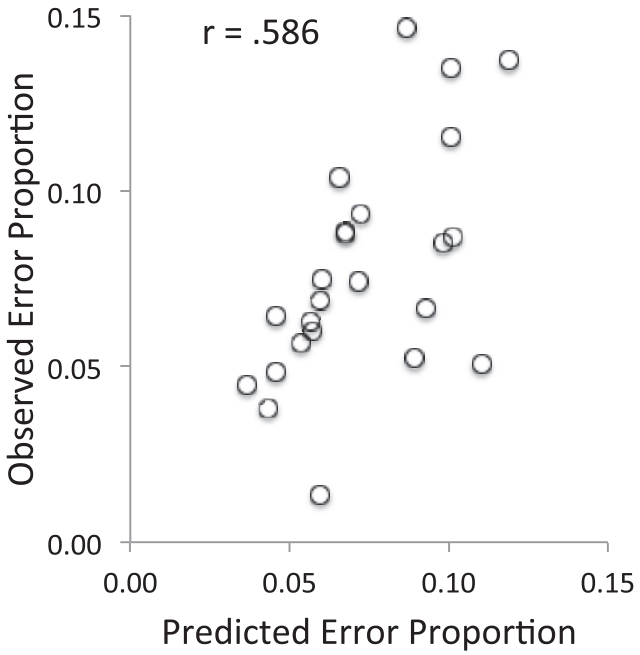

Figure 11. Predicting differences between typists: Correlation between error rates observed in typists and error rates produced by simulations with typists' best fitting parameters. Each point represents one of 24 typists.

To determine how well CRU predicts this benchmark in typing, I compared transposition gradients for observed and predicted typing performance. I calculated the transposition gradients for each typist's observed and simulated error corpus. I used the entire error corpus, including words with multiple errors. I focused on 5- to 8-letter words because they were as long as the lists typically studied in serial recall. I ignored correctly typed words in my calculations, which results in an underestimation of the probability of typing a letter in its correct position and an overestimation of the probability of typing a letter in another position. However, observed and predicted gradients are affected in the same way, so focusing on erroneous words will not bias the comparison between them. Excluding correctly typed words makes the trends easier to see.

I calculated transposition gradients for each word length separately and found they were very similar, so I collapsed across word length. The mean transposition gradients for observed and predicted typing performance are presented in Figure 12. The root mean squared deviation (*rmsd*) between the 64 observed and predicted proportions in the figure is *rmsd* = .064. The correlation $r = .960$. The fit is quite good given that only two parameters (β and S) were varied to fit individual keystrokes and no further parameters were adjusted to predict the transposition gradients.

Both the observed data and the simulated data show the locality constraint. Averaged over letters and positions in the word, the proportion of letters typed in correct, adjacent (± 1), and remote ($> \pm 1$) positions were 0.589, 0.220, and 0.008 in the observed data and 0.632, 0.192, and 0.013 in the simulated data. Figure 12 shows the fits were good for the middle letters in the word (4–6). CRU overpredicted the early letters (1–3) and underpredicted the later ones (7–8). In the observed data, the tendency to recall adjacent letters increased across letters in the word, broadening the transposition gradient. CRU captured this pattern in the data.

CRU's ability to predict transposition gradients that exhibit the locality constraint stems from the similarity structure in its stored contexts. Temporally adjacent contexts are more similar than temporally remote contexts, and so are more likely to be retrieved in error. This can be seen in the plots of dot products against position in the word in Figure 5. Adjacent dot products are higher than remote dot products. Higher dot products result in higher retrieval probabilities.

Serial Position Curves

Theories of serial and free recall address *serial position curves*, which plot the probability of recalling an item in its correct position. This information is contained in the transposition gradient, when item position equals reported position. I plotted observed and predicted serial position curves for typing in Figure 13. Again, the estimates include only words with errors. Including words typed correctly would increase the probabilities by the same amount for observed and predicted data. Both show a reduction in

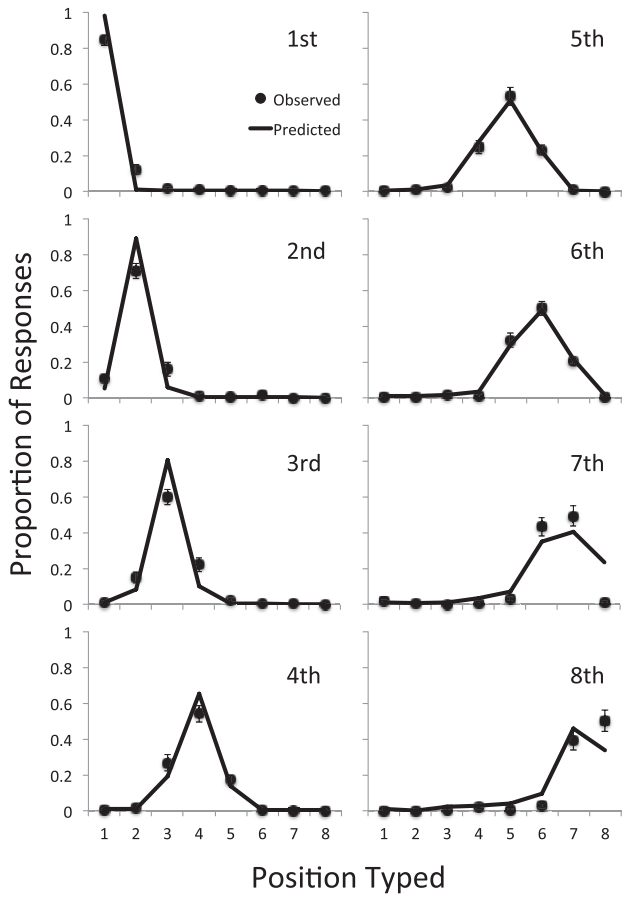

Figure 12. Observed and predicted transposition gradients, which plot the probability that a letter from the word is typed in each position in the word. Points are observed data. Lines are model predictions. Each panel represents a different letter from the word (1st-8th). Error bars are standard errors of the mean.

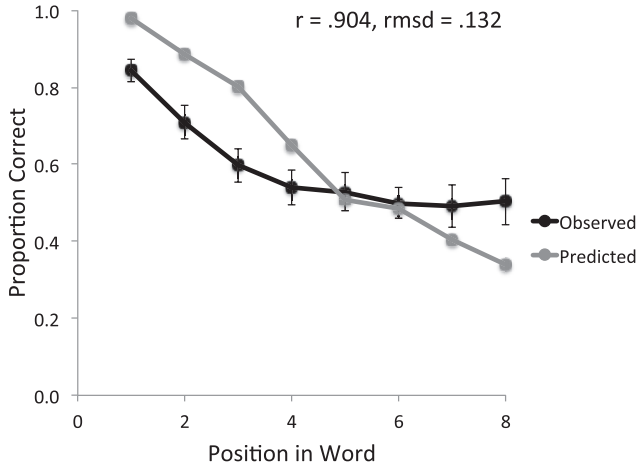

Figure 13. Observed and predicted serial position curves, which plot the probability that a letter is typed in its correct position in the word. Error bars are standard errors of the mean.

accuracy over serial position, as is typical of serial recall. The predicted curve overestimates the early serial positions and underestimates the later ones, like the transposition gradients.

It is interesting that CRU predicts a serial position curve with a strong primacy effect because there is nothing built into it to produce primacy. By contrast, many models of serial recall have to build in special mechanisms to account for primacy (for a review, see Lewandowsky & Farrell, 2008). I suspect the steepness of the serial position curve may be an artifact of the scoring method, which may be more appropriate for errors in serial recall than for errors in typing. Insertions (adding a letter, as in DIOGS for DOGS) and omissions (deleting a letter, as in DGS for DOGS) are common errors in typing but rare in serial recall. Often, participants in serial recall experiments are allowed to say "blank" or leave a position unfilled if they cannot remember an item, so subsequent items may be recalled in their correct serial positions. When typists make insertion and omission errors, every letter after the first error is scored as incorrect because it occurs one position later (insertions) or one position earlier (omissions) than it should have. This reduction in accuracy will affect the later part of the list more than the earlier part, producing the steep serial position curves in the observed and predicted typing data. The scoring artifact may also produce the broadening of the transposition gradient for later letters in the word (see Figure 12).

Serial Order: Lag Conditional Recall Probability Functions

Theories of free recall address the order in which items are recalled by evaluating *lag conditional recall probability functions*, which plot the probability of recalling an item from position $N \pm M$ given that the item in position N was recalled (Howard & Kahana, 2002; Lohnas et al., 2015; Polyn et al., 2009). If items are recalled in order, then $N + 1$ is the most likely transition. There is a tendency to recall items in order in free recall. The tendency should be much stronger in serial recall and typing.

I calculated lag conditional recall probability functions for each typist's observed and simulated error corpus. I used the entire error corpus including words with multiple errors and I focused on 5–8 letter words to have a range of ±4 lags. As before, I ignored words typed correctly, which would have a probability of 1 for lag $N + 1$ and a probability of 0 at all other lags. Excluding them underestimates the value for lag $N + 1$ and overestimates the value for other lags, but does not bias the comparison between observed and predicted functions. I calculated lag conditional recall probability functions separately for each typist and each word length. The functions for different word lengths were very similar, so I collapsed across word length. The means across typists for observed and predicted functions are plotted in Figure 14.

As in free recall, the lag conditional recall probability functions for typing are strongly asymmetrical. The probability of typing letter $N + 1$ after typing letter N is much higher than the probability of typing letter $N - 1$. This reflects the high level of accuracy in skilled typing. Letters tend to be typed in order. The model's predictions show the same asymmetrical pattern as the observed data. The *rmsd* between the 9 observed and predicted points in Figure 14 is 0.046 and the correlation is $r = .983$. The fit is good given that only two parameters (β and *S*) were varied to fit individual keystrokes and no further parameters were adjusted to predict the lag conditional recall probability functions.

In CRU, the asymmetrical lag conditional recall probability functions result from the associations between contexts and key locations (see Figure 2). Each context is associated with the key that follows it, so letter *N* is likely to be followed by letter $N + 1$. Observed and predicted functions also show a locality constraint, in that near lags (±1) have higher probability than far lags (>±1). This follows from the similarity structure in the stored contexts, where adjacent contexts are more similar than remote ones. CRU predicts a higher probability of immediate repetition (lag 0) than the observed data. This may suggest a tendency to inhibit the key that was just struck, which is a common feature in models of serial order but not implemented in CRU (see Response Suppression in the General Discussion).

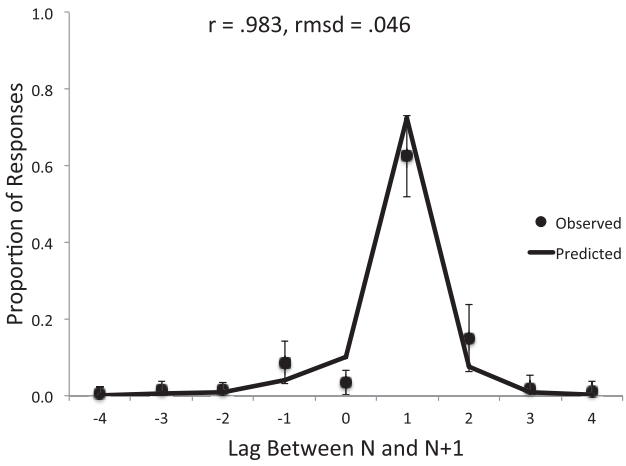

Figure 14. Observed and predicted lag conditional recall functions, which plot the probability of typing a letter *M* steps before or after typing letter *N*, where *M* ranges from −4 to + 4. Points are observed data. Lines are model predictions. Error bars are standard deviations.

Error Magnitude and Recovery From Errors

To estimate the magnitudes of the errors, I calculated *edit distance* between erroneous and correct strings. Edit distance is the number of editing operations (changes) required to transform one string into another. *Hamming distance* allows substitution as the only editing operation (Hamming, 1950). Elements of strings either match or mismatch. Hamming distance is essentially the strict scoring method used to calculate serial position curves. It does not capture the similarities between strings with insertions and deletions (DIOGS or DGS for DOGS). *Levenshtein distance* includes insertion and deletion as well as substitution (Levenshtein, 1966). *Damerau distance* includes transposition (DGOS for DOGS) as well as insertion, deletion, and substitution (Damerau, 1964). A transposition requires two steps in Levenshtein distance (two substitutions) but only one (transposition) in Damerau distance.

I calculated Levenshtein and Damerau distances for each typist's observed and simulated error corpus using Matlab algorithms provided by Schauerte and Fink (2010). The means across typists are plotted in Figure 15. Observed and predicted functions have similar shapes for both distance measures, peaking at distance = 1 and declining exponentially as distance increases. The fit between observed and predicted was better for Damerau distance (*rmsd* = 0.057, r = .992) than for Levenshtein distance (*rmsd* = 0.119, r = .922) because the model underpredicts the number of transposition errors (see below). Again, the distances were not fitted directly but were measured from simulations using the best fitting parameters from fits to individual keystrokes.

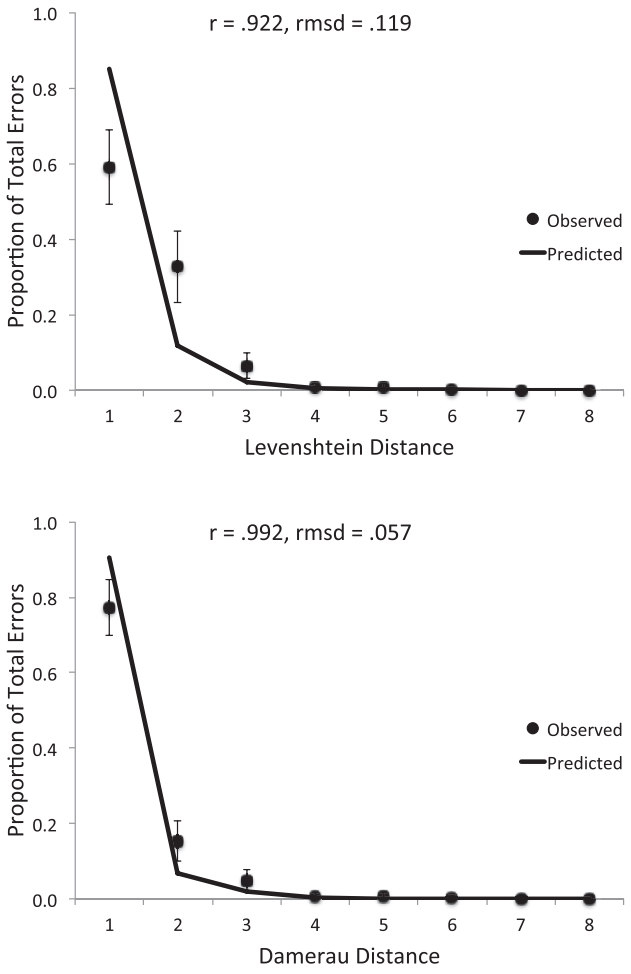

Figure 15. Predicted and observed distributions of Levenshtein and Damerau distances between correct and error responses. Error bars are standard deviations.

It is significant that the modal error distance was 1 for observed and predicted errors. The observed results indicate that typists usually made one error and recovered from it, finishing the rest of the word correctly. This might suggest that typists monitor their typing closely, engaging top-down control processes to detect and recover from errors (Crump & Logan, 2013; Logan & Crump, 2010). However, the model predictions show the same modal error distance of 1, recovering from errors with no top-down control. CRU recovers from errors because the similarity structure in the stored contexts exerts strong constraints on keystroke selection. Errors add noise to the context retrieval process but not enough to prevent retrieval of the correct response. According to CRU, experts type automatically without having to think about recovering from errors (Tzelgov, 1997, 1999).

Figure 16 illustrates how similarity allows automatic recovery from errors. Panel A shows dot products after typing DG (error) when intending to type DOGS. The highest dot product is for S, producing the appropriate continuation of the string following G. The second highest dot product is for O, producing the transposition DGO. Panel B shows dot products after recovering from the omission and typing DGS. The highest dot product is for typing the space bar to end the word, which is the appropriate continuation of the string following GS. Panel C shows dot products after making a transposition error and typing DGO. The highest dot product is for S, which would complete the string. Panel D shows dot products after making a substitution error and typing DI. The black line shows the case in which the substitution is copied into the current context (when it is a context retrieval error). The most likely continuation is DIO, which is the letter that follows D in the correct sequence. The dashed gray line shows the case in which the substitution is not copied into the current context (when it is a finger selection error). The letter I has been typed but O is copied into the current context, so the most likely continuation is DOG.

In all cases, recovery from the error is the most likely outcome. CRU is robust to errors because the similarities among the stored contexts exert strong constraints on the choice of letters. CRU recovers from errors automatically without any special mechanisms or top-down control (Tzelgov, 1997, 1999). I discuss how it might be extended to address explicit error detection and correction below (Top-Down Control).

Error Patterns

Model based error categories. Errors invite taxonomies. My theory provides a taxonomy for errors based on the source of the error in the model. It predicts two main categories of errors: *Errors from the same word* are produced by context retrieval; *Errors from neighboring keys* are produced by finger selection. Combining these error categories factorially produces four categories. I categorized each typist's observed and predicted errors in terms of this taxonomy, focusing on single and transposition errors (Damerau distance = 1) in 3- to 8-letter words. Neighboring keys were

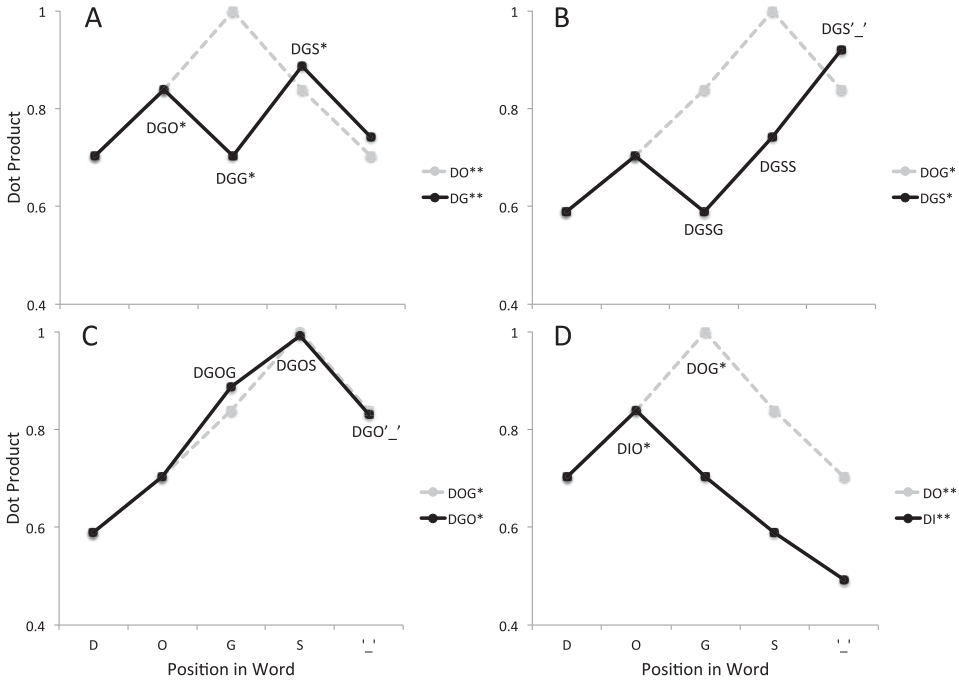

Figure 16. Recovery from errors. Panel A: Dot products for different positions in the word after typing DO (correct) or DG (error). Panel B: Dot products after typing DOG (correct) and DGS (omission error). Panel C: Dot products after typing DOG (correct) and DGO (transposition error). Panel D: Dot products after typing DI (substitution error that is included in the updated context) and DO (substitution error that is not included in the updated context). Black lines represent dot products after an error (typing DG when intending to type DOGS). Gray dashed lines represent dot products after a correct response (typing DO when intending to type DOGS). Letters in the legend represent the letters typed so far. The letters next to the data points represent choices of the next letter (DGS* represents typing S after DG).

defined as those horizontally, vertically, or diagonally adjacent to the intended key. The number of neighboring keys ranged from 2 (P) to 8 (e.g., G) with an average of 5.3. The mean error proportions across typists are presented in Figure 17.

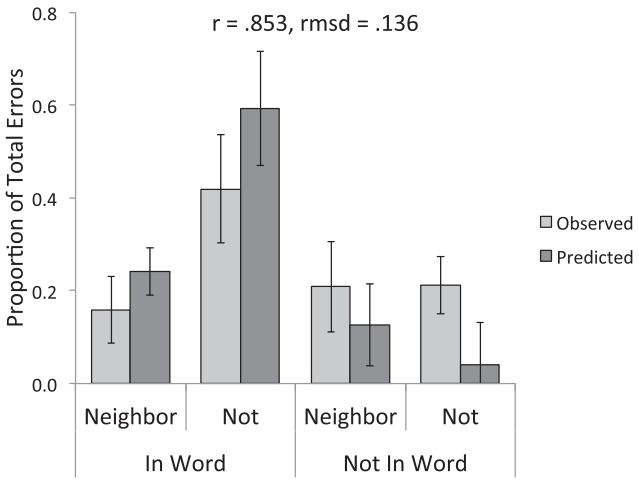

Figure 17. Observed and predicted proportions of total errors from words or neighboring keys. Error bars are standard deviations.

The distributions of observed and predicted error proportions were similar with some notable differences. CRU produced many fewer errors that were not in the word and not neighbors than typists did, but this was by design. CRU can only produce non-neighbor errors if a remote key wins the finger selection race, and this rarely happens (see Figure 8). I assume that typists produce non-neighbor errors with mechanisms outside the scope of the model (e.g., misaligning their fingers with the keyboard). Like the typists, CRU produced more errors "in word" than "not in word," reflecting the balance between β in context retrieval and *S* in finger selection. Again, the parameters were chosen to maximize the likelihood of individual keystrokes, not error categories. The fit may be improved by adjusting parameters.

Traditional error categories. Studies of serial order often distinguish between *omissions* (typing DG for DOG), *transpositions* (typing DGO for DOG), *insertions* (typing DIOG for DOG), and *substitutions* (typing DLG for DOG; see Dell, 1986; Norman, 1981; F.A Logan, 1999; Reason, 1990; Salthouse, 1986). I categorized the errors in each typist's observed and simulated corpus in terms of this taxonomy, restricting my analysis to 3- to 8-letter words with single errors or transpositions (i.e., errors with Damerau distance = 1). I categorized errors with the algorithm in Table 2, which uses the lengths of the correct and error strings and the Damerau and Levenshtein distance between them. The algorithm agrees almost perfectly with human categorization of these errors.

Table 2
Rules for Categorizing Single Errors and Transpositions

```
IF Damerau = 1 THEN
   IF Elength < Clength THEN Omission
   IF Elength > Clength THEN Insertion
   IF Elength = Clength AND Levenshtein = Damerau THEN
      Substitution
   IF Elength = Clength AND Levenshtein = Damerau + 1 THEN
      Transposition
END
```

Note. Errors with Damerau distance = 1; Damerau = Damerau distance; Levenshtein = Levenshtein distance; Elength = Length of error string; Clength = Length of correct word.

The mean observed and predicted proportions of omission, transposition, insertion, and substitution errors are presented in Figure 18. Like the typists, CRU produced errors of each type. CRU overestimated the proportion of omission errors and underestimated the proportion of transposition errors. It may be possible to improve the fits by adjusting parameters. For example, decreasing β to give more weight to the past will increase the frequency of transposition errors (see Figure 6). However, the model was not fitted to distributions of errors directly. The parameters were chosen to maximize the likelihood of individual keystrokes.

Traditional error categories are defined by happens after the error instead of what causes it. Omissions and transpositions both start by skipping a letter (typing DG for DOGS). They are classified as omissions if the letter following the error continues the sequence (DGS) and transpositions if the letter following the error is the omitted letter (DGO). The factors that cause the initial error may be the same. The differences may emerge only in the process of recovering from errors. In CRU, omissions and transpositions have the same cause. They are retrieved by the same current context (DG) with probabilities that depend on similarities to stored contexts (see Figure 16A–16C).

Substitutions and insertions both start by typing a letter that does not belong in that place in the word (Figure 16D). They are classified as substitutions if the error string is as long as the correct string and as insertions if the error string is longer (see Table 2). In CRU, substitutions are likely to occur when the erroneous letter is not copied into the current context, as in adjacent neighbor errors made in finger selection. Context retrieval "thinks" the correct letter has been typed and so selects the next one in the sequence (Figure 16D, dashed gray line). Insertions are likely to occur when the erroneous letter is copied into the current context, as in a context retrieval error. It adds noise to context retrieval, but the letter following the preomission letter is chosen most often (Figure 16D, black line). Consistent with this prediction, 62.4% of substitutions were adjacent neighbors but only 43.6% of insertions were.

Double letter errors. Following Lashley (1951), Rumelhart and Norman (1982) drew attention to double letter errors (WEEL for WELL), which they viewed as a critical test of their theory. They assumed typing is controlled by interactive activation in a set of schemas. A word schema activates letter schemas, which activate motor schemas that specify hand, finger, and movement (see Figure 19). The motor schemas feed activation back to the letter schemas. Serial order is controlled by *successor inhibition*, implemented as inhibitory connections between letter schemas. Each schema inhibits every schema that follows it, which produces a gradient of activation that favors the earliest schema in the sequence (Bryden, 1967; Estes, 1972). When a keystroke is executed, the schema for the letter is inhibited so the next letter in the sequence can win the competition.

Rumelhart and Norman (1982) assumed a "pure type" representation, in which there is one schema for each letter, so the model can only represent unique sequences of letters. Words with repeated letters (METE) are planned up to the repeated letter (MET—E). The repeated letters are planned separately. Doubled letters (MEET) are represented by attaching a "double" schema to the schema for the doubled letter (M E-double T). Sometimes the double schema is applied to the wrong letter, producing a double error (MMET for MEET). Rumelhart and Norman found double errors in a corpus of errors and interpreted them as support for their theory.

The "double schema" hypothesis predicts double errors in words with doubled letters (MEET) because the doubled letters activate the

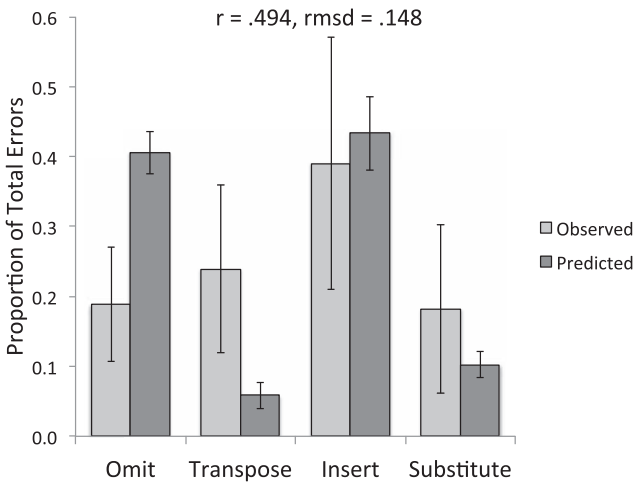

Figure 18. Observed and predicted proportions of total errors in traditional error categories. Error bars are standard deviations.

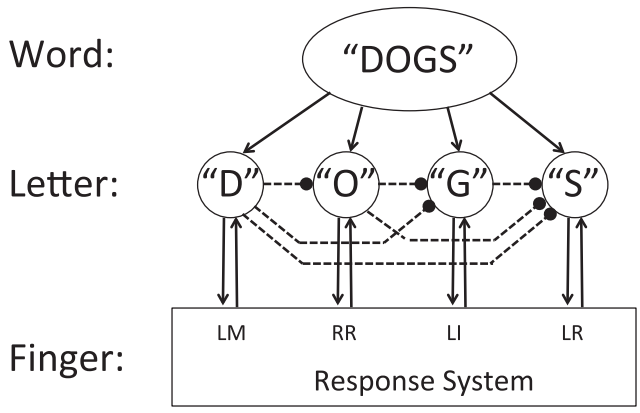

Figure 19. The Rumelhart and Norman (1982) model of typing. Solid lines and arrows represent activation. Dashed lines and dots represent inhibition. DOGS is activated in the word level. Activation spreads to the letter level, from the letter level to the finger level, and from the finger level back to the letter level. Each letter inhibits every letter that follows it, producing a gradient of activation that sequences responses.

double schema, which is attached to the wrong letter. Words with unique letters (MEAT) or repeated letters (METE) should not produce double errors because there are no doubled letters to activate the double schema. I tested this prediction in the error corpora from the 24 typists, counting the frequencies of double letter errors in words with unique, repeated, and doubled letters. On average, typists made 4.96 (SD = 3.68) double letter errors, which is 4.24% of their total errors. I calculated the proportions of double letter errors that came from unique, repeated, and doubled letter words. The average proportions across typists are plotted in Figure 20.

Contrary to Rumelhart and Norman's (1982) prediction, double letter errors were least likely in doubled letter words and most likely in unique letter words, where the double schema should not have been activated. However, unique letter words were more common than doubled letter words in the corpus so there were more opportunities for errors. The relative frequencies from the corpus are plotted as the lightest bars in Figure 20. The observed proportion for doubled letter words was higher than the proportion in the corpus, and the observed proportion for unique letter words was lower than in the corpus. This may suggest that the double schema may have been activated more often in doubled letter words than in unique words.

CRU requires no special mechanism to type doubled or repeated letters. Each letter command is represented the same way whether or not it is repeated, as a unit vector with 1 in the element corresponding to the letter and 0 in all other elements. The letter commands are used to update the current context following Equation 1. Contexts with doubled letters are built by adding two vectors that represent the same letter command to the current context. Contexts with repeated letters are built by adding the second vector with some lag. The resulting similarities among contexts in words with unique, repeated, and doubled letters are plotted in Figure 21. Similarity is highest for the letter that is most appropriate for the context even in words with doubled letters, so

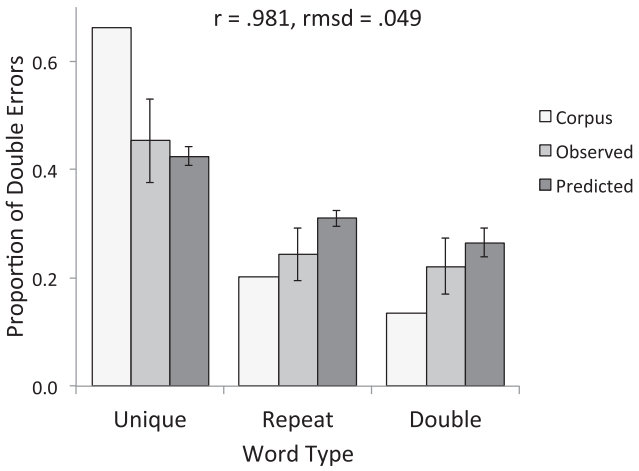

Figure 20. Proportions of double errors from words with unique (MEAT), repeated (METE), and doubled (MEET) letters. The lightest bars represent the relative frequencies of unique, repeated, and doubled letter words in the corpus. The intermediate gray bars represent the observed proportions of doubled errors averaged over typists. The dark gray bars represent the predicted proportions of doubled errors from the simulation, averaged over typists. Error bars are standard errors of the mean.

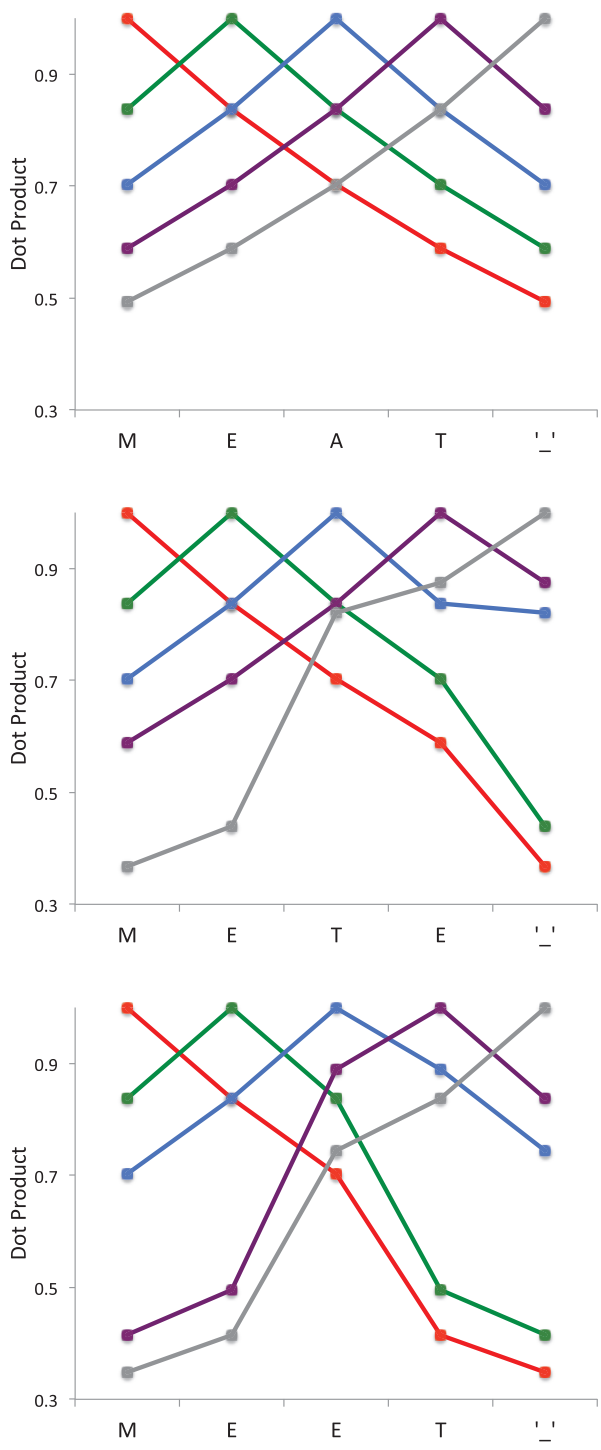

Figure 21. Similarities among stored contexts for words with unique (MEAT), repeated (METE), and doubled (MEET) letters. The points above each letter represent the competition.

CRU types the correct sequence most often. Adding two copies of the same letter command distorts the similarities and increases competition among contexts in double and repeated letter words, but the correct sequence is most likely nevertheless. Equation 1

blends successive contexts so that the context for the first letter in a double is different from the context for the second (M vs. ME in MEET). The same is true for repeated letters (M vs. MET in METE). Thus, each letter has a unique context even when letters are doubled or repeated.

CRU makes quantitative predictions about the relative frequencies of double letter errors in words with unique, repeated, and doubled letters. These predictions are summary statistics calculated on the simulated error corpora for each typist. They were not fit directly. I calculated the frequencies of double letter errors in the simulated error corpus for each typist, computed proportions, and plotted the averages across typists in Figure 22. The proportions predicted by the model are very close to the observed proportions, showing the same ordering of conditions. Thus, CRU accounts for double letter errors without a double schema. The double schema is not necessary.

CRU makes quantitative predictions for overall error rate for words with unique, repeated, and doubled letters. Those predictions are plotted along with the observed values in Figure 22. CRU underpredicts error rate for unique letter words and overpredicts error rate for doubled letters, showing substantial effects of word type whereas the observed data show small effects. It may be possible to improve the fit by increasing β to steepen the similarity gradients. β was not adjusted to fit these error proportions. It may also be possible to improve the fit by allowing repeated and doubled letter words to have higher thresholds than unique letter words. Future research will determine whether changes in β and threshold can reduce the differences between unique, repeated and doubled letter words.

Toward a Theory of Typing

The goal of this article is to develop and test a theory of automatic control in skilled typing. I focused on serial order and finger choice because they are the things that must be controlled automatically. CRU is a model of automatic control, but it is not yet a complete theory of typing. It does not address keystroke timing, which provides much of the data a theory of typing must account for, and it does not address top-down control, other than saying what is required to start typing a word and what signals the end of a word. Here I suggest ways to extend CRU to become a complete theory of typing.

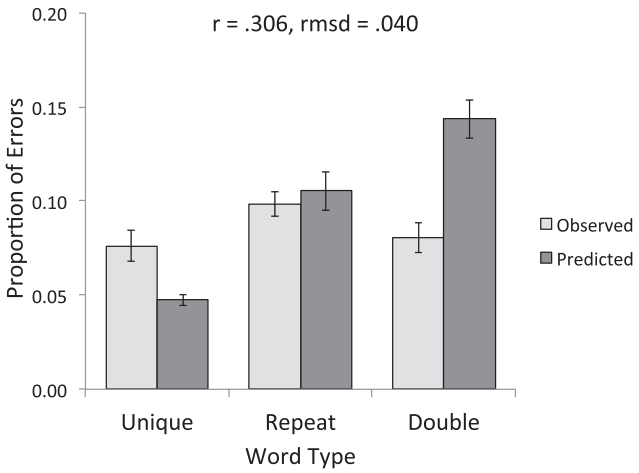

Figure 22. Observed and predicted error proportions for words with unique, repeated, and doubled letters. Error bars are standard errors of the mean.

Keystroke Timing

Distributions of interkeystroke intervals. Equation 5 predicts latency as well as probability. In principle, it could be used to fit distributions of interkeystroke intervals. However, fitting distributions is impractical. Interkeystroke interval is the sum of the durations of context retrieval, finger retrieval, and movement, so its distribution is the convolution of three distributions, one for each component. Specifically, the distribution would be the convolution of Equation 5 for context retrieval, Equation 5 for finger retrieval, and an unknown distribution of movement times. Even without movement times, it would be difficult to isolate the parameters for context retrieval and finger retrieval in the convolution, separately identifying drift rates and thresholds for each stage. Models of responses time distributions assume a single decision stage (e.g., Ratcliff & Smith, 2004; Teodorescu & Usher, 2013). No one has tried to fit distributions of two successive (serial) decision stages. That is an important goal for future research.

Moreover, movement times would likely dominate the convolved distributions. Movement times are typically longer than interkeystroke intervals (Flanders & Soechting, 1992; Soechting & Flanders, 1992) and movement-related factors have large effects on keystroke timing (Salthouse, 1986). Systematic and random variability would have to be separated, ideally with a model. That too is an important goal for future research.

Variability in interkeystroke interval. In its current form, the model underpredicts variance in interkeystroke interval. I simulated typing of all 3- to 8-letter words in all four paragraphs using the best fitting parameters ($\beta = .5457$ and $S = .1956$) and found a mean and standard deviation of 700 and 26 for correct responses and 717 and 33 for errors (units are arbitrary). The coefficient of variation (standard deviation divided by the mean) was 0.037 for correct responses and 0.046 for errors.

In the observed data, mean and standard deviation of interkeystroke intervals were 144 ms and 96 ms, respectively. The coefficient of variation is 0.667, which is much larger than in the model. The observed data contain systematic and random motor variance. Interkeystroke interval varied systematically with movement distance. It was 148, 146, and 189 ms for top, home, and bottom row keys, respectively. Interkeystroke interval also varied systematically with the transition between fingers required for successive keystrokes. Between-hand transitions (129 ms) were faster than same-hand transitions (154 ms). Same key transitions were slower (162 ms), and same finger transitions were slowest (196 ms; also see Salthouse, 1986).

To separate systematic and random variability, I sorted interkeystroke intervals by transitions between successive letters and calculated means and standard deviations for each transition (see Appendix C, Tables C1–C3). With the systematic motor variance removed, the mean interkeystroke interval was 146 ms and the mean standard deviation was 49, for a mean coefficient of variation of 0.333, reducing the original value by half. The coefficient of variation for same key transitions (repeated keystrokes), which should have the smallest motor variability, was only 32/162 =

0.195. (Coefficients of variation for same-finger, same-hand, and different-hand transitions were 0.215, 0.316, and 0.396, respectively.) The smallest coefficient of variation, for typing P twice, was 15/154 = 0.097. Compared with these values, the model values (0.037 and 0.046) may not be inordinately small.

Central tendency in interkeystroke interval. CRU provides four ways to account for central tendencies of keystroke timing effects: Changes in drift rate, changes in threshold, competition from letters in other words or locations, and learning. In the current version, the maximum drift rate is fixed at 1, which is the normalized length of context vectors, so predicted retrieval time is approximately threshold/1. The drift rate could be scaled into the range of observed interkeystroke intervals. Threshold could be raised or lowered to manage the speed–accuracy trade-off or to anticipate difficult or easy retrievals (Yamaguchi, Crump, & Logan, 2013). CRU already includes the possibility of interference from other words (see Figure 7). The current version assumes the diffusions race independently, so the effect appears in accuracy rather than timing. It may be possible to implement competition within an independent race architecture by dividing each drift rate by the sum of the drift rates (Teodorescu & Usher, 2013), which could be interpreted as reflecting inhibition (Lo & Wang, 2006) or capacity limitations (Logan, 2002). CRU provisionally assumes perfect learning, which is unlikely. Given the instance-like nature of stored context representations and their associations to responses, storing instances would be a natural way to represent learning (Logan, 1988).

Words versus nonwords. CRU predicts that nonwords will be typed more slowly than words (Fendrick, 1937; Hershman & Hillix, 1965; Salthouse, 1986; Yamaguchi & Logan, 2014b, 2016). Unfamiliar nonwords must be typed one letter at a time with serial order controlled in the outer loop. CRU would represent each letter with two stored contexts, one to initiate typing ("Type A" and "motor blank" retrieves "A") and one to indicate typing is finished ("Type A" and "motor A" retrieves "space"). Thus, typing an *N*-letter nonword requires 2*N* retrieval operations, whereas typing an *N*-letter word only requires $N + 1$ retrieval operations. Alternatively, the outer loop may rely on visual feedback from the screen or fingers as a signal to initiate the next keystroke. In this case, nonwords would be typed more slowly because the time to process sensory feedback is longer than interkeystroke interval (Lashley, 1951). CRU does not yet explain why nonwords that resemble words are typed faster than random strings of letters or how new words are learned. These are important questions for future research.

Initial latency. Figure 23 plots interkeystroke interval as a function of position in the word for 3–7 letter words in the corpora. First letter keystrokes were 57 ms longer than the mean interkeystroke interval for subsequent keystrokes, $F(1, 23) = 10.39$, $p < .001$, $MSE = 377.0$, $\eta_p^2 = 0.856$. CRU predicts that the first keystroke will take longer than subsequent keystrokes because it includes the time for top-down processes to generate a word to be typed and initialize the current context with a word command that represents the word. In discrete typing tasks, in which typists have to type single words presented one at a time, first keystroke latency includes perceiving the word and retrieving lexical information about it as well as the time to select keys and fingers. In continuous typing, the outer loop has to select the word to type next before selecting keys and fingers. CRU does not model those outer loop processes, so it cannot predict their duration precisely. However, it must take time to generate a word command vector from a lexical representation of the word in the outer loop, so first letters should have longer latencies (Logan & Gordon, 2001).

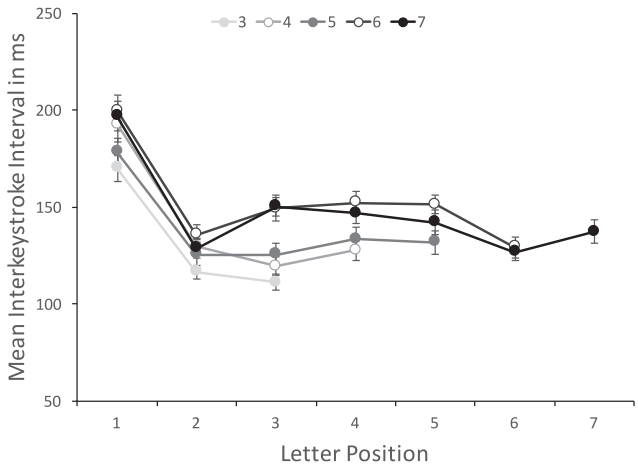

Figure 23. Mean interkeystroke interval for correctly typed 3- to 7-letter words as a function of position in the word. Error bars are standard errors of the mean.

Planning time. Motor planning is reflected in the effect of sequence length on the time to initiate a sequence of movements, in which longer sequences take longer to initiate (Henry & Rogers, 1960; Rosenbaum, Kenny, & Derr, 1983). Similar sequence-length effects have been observed in typing (Sternberg, Monsell, Knoll, & Wright, 1978). CRU does not predict planning time effects, as it assumes no advance planning, and words of all lengths are represented as single-element vectors. CRU models experts typing familiar words. Studies showing planning time effects use unfamiliar materials (nonwords in Sternberg et al., 1978) and low levels of practice, compared with the years of experience of skilled typists. Yamaguchi and Logan found stronger sequence length effects for typing nonwords than for typing words in both initial latency and interkeystroke interval (Yamaguchi & Logan, 2014b, 2016; Yamaguchi, Logan, & Li, 2013), suggesting that unfamiliar materials require planning but familiar materials do not.

The present data, plotted in Figure 23, show a 6-ms/letter increase in first-keystroke latency with word length, suggesting a planning effect in skilled typists. However, interkeystroke intervals for subsequent keystrokes show the same 6 ms/letter increase, which suggests an effect on typing the whole word instead of an effect on planning. Indeed, word length was negatively correlated with word frequency in the copora. The mean Kučera and Francis (1967) frequencies were 14587, 1865, 605, 135, and 56 per million for 3- to 7-letter words, respectively. Experiments on discrete typing with frequency controlled show smaller word length effects, particularly for keystrokes beyond the first (Yamaguchi & Logan, 2014b, 2016; Yamaguchi et al., 2013).

Slow errors. CRU predicts errors will be slower than correct keystrokes. In the simulations described above, errors took 717 arbitrary units while correct responses took 700. This is a general property of race models of choice, unless there is substantial between-trial variability in threshold or starting point (Logan et al., 2014; Ratcliff & Smith, 2004; Teodorescu & Usher, 2013). The

drift rate for correct responses is higher than the drift rate for errors, so errors will be slower than correct responses. This can be seen in the plots of the dot products for adjacent contexts (see Figure 5). The dot products for errors are always lower than the dot products for correct responses, so errors should be slower than correct responses. This prediction is confirmed in published data: errors are usually slower than correct keystrokes (Crump & Logan, 2013; Logan & Crump, 2010; Salthouse, 1984; Yamaguchi et al., 2013).

Post error slowing. CRU predicts a modest amount of posterror slowing. Typists often slow dramatically after errors, by as much as 150 ms (Logan & Crump, 2010; Salthouse & Saults, 1987). Much of the slowing is strategic. Typists stop slowing after errors when speed is stressed (Yamaguchi et al., 2013). The experimenter's instruction not to correct errors may produce slowing, as typists must inhibit their automatic tendency to do so (Crump & Logan, 2013). CRU predicts posterror slowing because the posterror context includes the error and so does not match any of the correct stored contexts exactly. Thus, drift rates will be lower following errors. This can be seen in the posterror dot products in Figure 16. The posterror dot products are about as large as the dot products that produced errors, and so should produce about the same modest slowing as errors do.

Word and bigram frequency. Words and letter sequences that occur more frequently in the language are typed faster (Salthouse, 1986). CRU may already have the capacity to predict bigram frequency effects. The motor part of the stored context vectors represents the letters in the words on which CRU was trained. Letter sequences that occur in more words will be represented more often in the set of stored context vectors, as there is one context for each letter in each word (plus one for the space bar). Frequent motor contexts increase the number of racing diffusions that produce the same response—CRU may retrieve the right letter from another word with the same letter sequence—and that will speed retrieval time (Logan, 1988). Word frequency effects could be accounted for similarly, by repeating words in the training corpus in proportion to their frequency in the language, as is done in models of reading. The more runners in the race, the faster the retrieval (Logan, 1988).

Top-Down Control

CRU models automatic control in the inner loop. A complete theory of typing must address control in the outer loop, saying what it controls and how it does so. Some of that control may be automatic. Typing is an expression of language and there are many automatisms in comprehending and producing language. Some control may be less automatic, adapting to special circumstances and novel goals (Logan, 2017).

The outer loop controls parameters of the inner loop, like the threshold in the racing diffusions (Logan & Gordon, 2001). Typists adjust their typing rate in response to speed-accuracy instructions (Yamaguchi et al., 2013) and changes in feedback (Snyder, Logan, & Yamaguchi, 2015). The outer and inner loops rely on different feedback. The outer loop monitors visual feedback—the letters echoed on the screen (Logan & Crump, 2010). The inner loop monitors haptic and kinesthetic feedback—the feel of the keys as they are struck (Crump & Logan, 2010c). A complete theory of typing must specify the processes that use this feedback to initiate these changes. CRU provides a theory of the inner loop, on which these higher-level processes may be grounded (Logan & Gordon, 2001).

Punctuation and capitalization. CRU models typing strings of lower-case words separated by spaces. Punctuation and capitalization require departures from this habitual typing mode to insert changes dictated by sentence and phrase structures. One way to deal with them is through top-down control. To capitalize "most," the outer loop might send a single letter command for typing "capital m," and let CRU carry on from there. "Capital m" might be represented by setting two elements in the letter command vector, one representing "m" as in lower case typing and one representing the shift key. This context would match the initial context of "most" except for the shift key, so it should trigger retrieval of the rest of the letters in sequence.

Top-down control of punctuation is more of a challenge. Punctuation comes at the ends of words, and CRU signals the ends of words by striking the space bar and sending a copy of the motor command to the outer loop. The outer loop could monitor the keystrokes in words it intended to punctuate, and insert the punctuation mark at the right point, but monitoring individual keystrokes slows typing substantially (Logan & Crump, 2009; Snyder & Logan, 2013). Alternatively, the outer loop could inhibit the space bar command and insert a command for the punctuation. This would be hard to do reactively, because interkeystroke intervals are often around 150 ms (Salthouse, 1986) and RTs to stop signals (like the command for the space bar) are longer than that in skilled typists (Logan, 1982; Salthouse, 1984). The outer loop could do it proactively, substituting the punctuation mark for the space bar, but typing is much slower when typists substitute letters (Yamaguchi & Logan, 2014a).

In principle, CRU could model automatic punctuation and capitalization in the inner loop by assuming a different word representation and a different set of stored contexts for capitalized and lower case words and for words followed by spaces and words followed by punctuation. This would require twice as many stored contexts to represent capitals and six times as many stored contexts to represent punctuation (i.e., "." "," "?" ":" ";" "!"). It may be possible to reduce this number by developing CRU's assumptions about similarities among word commands. CRU's word commands are orthogonal (dot product = 0) because each word is represented by a different single element (see Figure 7). Word commands for capitalized and punctuated words might overlap with word commands for lower case versions and take advantage of their structure. So might word commands for inflections ("move," "moved," "moving"). Overlap with word contexts might also explain why word-like nonwords are typed faster than random strings of letters. Exploring the similarity structure of word representations is an important topic for future research.

Error detection and correction. CRU models how typists recover from errors. It does not model how they detect and correct them. Typists detect errors in different ways. Explicit error detection depends on the information echoed on the screen (Logan & Crump, 2010). Typists explicitly detect ~90% of their errors when their typing is echoed on the screen and ~60% when it is not (Snyder et al., 2015). Thus, visual feedback is important but haptic and kinesthetic feedback are also important. Some components of error detection may be automatic. Seeing an incorrect letter on the screen may automatically trigger a backspace response (Crump &

Logan, 2013), but the number of backspaces depends on the location of the error and how the typist decides to correct it (going back to the error vs. retyping the whole word). Intervals between backspaces increase before the correction, and the first two interkeystroke intervals of the correction are elevated (Crump & Logan, 2013), suggesting a top-down intervention (Yamaguchi & Logan, 2014a).

Correction would seem to require top-down control. Retyping the word would require reinstating the initial word command and letting CRU take over. Starting the correction at the error would require establishing the prior motor contexts somehow before CRU could take over. The outer loop might issue one or two letter commands to get the sequence started, as CRU depends more heavily on the most recent keystrokes. Developing a theory of top-down control is an important priority. CRU provides a model of the inner loop on which the theory may be grounded.

General Discussion

How do experts act without thinking? How can experts do more but think less than novices? My resolution to the paradox of expertise is the CRU theory of automatic control, which says what the inner loop controls (key choice, finger selection, serial order) and how it exerts control (context retrieval and updating). The theory fleshes out the idea that expert control is hierarchical, with one thought in the outer loop producing many actions in the inner loop. CRU assumes that one word from the outer loop instigates a series of key and finger retrievals by setting the initial context, which then evolves without top-down control until the word is finished (see Figure 2). Control is automatic in that it occurs without top-down monitoring (Tzelgov, 1997, 1999).

CRU accounts for the phenomena of automatic control. It predicts a lower load in working memory and a higher load in the motor system for skilled typing (see Figure 1). Skilled typists have one word active in working memory (Yamaguchi & Logan, 2014b) and that activates several letters in the motor system (Behmer & Crump, 2017; Crump & Logan, 2010b). CRU predicts the importance of words in skilled typing. Stored contexts are built around words. The word command is common to all the stored contexts that express the word and selectively activates those contexts (see Figure 7). CRU has the capacity to represent large numbers of words. The simulations of the 24 typists used 47–65 words and could handle much larger vocabularies. CRU puts knowledge of key locations and finger-to-key mappings in the inner loop, in context retrieval and finger selection, respectively. It predicts that skilled typing can occur without top-down knowledge of key locations and finger mappings.

I evaluated the model by fitting it to error corpora from 24 skilled typists and predicting error frequencies, magnitudes, and patterns. The fits maximized the likelihood of single keystrokes in the error corpus using two free parameters (β and S). I generated predictions by simulating the model with the best fitting parameters and calculating summary statistics. For each typist, one set of parameters generated all of the predicted summary statistics without further adjustment (see Figures 12–15, 17, 18, 29, 22). In each case, CRU showed the same qualitative trends as typists and often fit the data quantitatively.

Implications for Theories of Automaticity, Skill, and Hierarchical Control

Automatic control. Perhaps CRU's strongest implication is its fundamental assumption that automatic processing is controlled. I have argued for automatic control since the 1980s (Logan, 1985, 1988; also see Tzelgov, 1997, 1999), but CRU makes the idea explicit computationally and algorithmically (Marr, 1982). Automatic control runs against the grain of much research on automaticity. Seminal theories like Shiffrin and Schneider's (1977) and Jacoby's (1991) contrast automatic and controlled processing as if they were opposites. Studies of Stroop, flanker, and priming effects often pit automatic processes against controlled processes. More recent research on task switching often focuses on overcoming automatic tendencies to use prior task sets (Vandierendonck, Liefooghe, & Verbruggen, 2010). Automatic and controlled processing became System 1 and System 2 in judgment and decision making research, where many studies pit System 1 against System 2 (Kahneman, 2011). Conflicts between automatic and controlled processing provide crucial data that any theory of automaticity must account for, but they draw attention away from situations in which automaticity and control go hand in hand, as in the many skills we rely on in daily life. Theories that explain how to overcome automatic processes do not necessarily explain how to harness them to fulfill intentions and they do not explain how that control can be automatic. Automatic control is a problem the field must solve. CRU offers one solution. I hope that many more will follow.

How is typing automatic? CRU has implications for how automaticity should be assessed. It recommends assessment by identifying the mechanism (Logan, 1988) over assessment of empirical properties (Logan, 1985; Moors & De Houwer, 2006). Typing is clearly automatic from the mechanistic perspective. CRU types automatically because typing is based on memory retrieval (context retrieval) rather than algorithmic outer-loop computation (hunting and pecking; Logan, 1988). CRU types automatically because its memory mechanisms choose correct sequences of keystrokes without top-down monitoring and control (Tzelgov, 1997, 1999). If CRU fits the data, then typing is automatic in these senses. The fit can be assessed from other aspects of behavior (e.g., sequences of keystrokes, transposition gradients, etc.), so the reasoning is not circular.

Typing is less clearly automatic from the perspective of its empirical properties. Consider three major properties of automaticity: speed, effortlessness, and autonomy (Logan, 1985; Moors & De Houwer, 2006). Typing meets the speed criterion because it is very fast. Effortlessness is less clear. Measured as dual task interference, typing is both effortless and effortful. Shaffer (1975) found little interference between typing visual text and shadowing auditory sequences, suggesting typing is effortless. Yamaguchi and Logan (2014b, 2016) found less interference with a concurrent memory load when typing words than nonwords, suggesting skilled typing is less effortful than unskilled typing. Yamaguchi et al. (2013) used the psychological refractory period procedure to control timing more precisely and found substantial dual task interference between typing and a concurrent tone discrimination task. This suggests typing is effortful.

CRU makes sense of these results. The shadowing task and the concurrent memory load interfere with outer loop processing, at the level of words. Executing all the keystrokes takes time, so the

outer loop often has to wait until the current word is finished before it sends the next one to the inner loop. This allows flexible timing in the outer loop, which may help reduce interference with shadowing (Broadbent, 1982; Pashler & Johnston, 1989). Typists think in words, so words will interfere less with concurrent memory loads than letter strings of the same length (Yamaguchi & Logan, 2014b, 2016). The psychological refractory period effect may occur in the inner loop (Yamaguchi et al., 2013). Context retrieval and key selection map conceptually onto decision and response selection stages in models of dual-task performance (Logan & Gordon, 2001; Meyer & Kieras, 1997; Pashler & Johnston, 1989). Interference in those stages is very hard to eliminate with practice.

The autonomy criterion is also unclear. Typing is autonomous in that it can go on to completion without top down control (Zbrodoff & Logan, 1986). Concurrent memory load studies support this assertion (Yamaguchi & Logan, 2014b, 2016). But typing is not autonomous in the sense that it is obligatory or ballistic. Skilled typists can readily stop typing in midword in response to an error (Crump & Logan, 2013) or a stop signal (Logan, 1982; Salthouse & Saults, 1987). CRU makes sense of these results as well. Usually, words will be typed to completion automatically, but automatic control can be interrupted with a top down signal that clears the word context or inhibits the growth of activation in the racing diffusions (Boucher, Palmeri, Logan, & Schall, 2007; Logan, Yamaguchi, Schall, & Palmeri, 2015).

The larger point is the value of viewing automaticity as something to be explained and not as an explanation in itself (Reynolds & Besner, 2006). To say that automatic processing is fast is not to say why it is fast, and why is an important question.

Hierarchical skills. CRU should generalize to other hierarchical skills in which one thought leads to many actions. CRU addresses a computational problem that is common to many skills—selecting targets for action, choosing effectors to act on the targets, and ordering the series of choices. It provides an algorithm that solves the problem using context retrieval and updating (Marr, 1982), which can be applied to other domains in which actions create the context in which subsequent actions are chosen. For example, CRU can account for nonstandard typing by reducing the number of fingers used (<8) and changing the mapping of fingers to keys. It could account for texting by exchanging 8 fingers for 2 thumbs. It could account for guitar playing by increasing the number of targets (there are 120–144 notes on a guitar neck) and increasing the number of postures for striking them. In each case, updating current contexts with the latest motor command would change its similarity to the stored contexts, and that could drive the required sequence of retrievals.

The literature is divided on whether hierarchical structure in behavior should be modeled with hierarchical processes (Cooper & Shallice, 2000; Rumelhart & Norman, 1982) or hierarchical representations with nonhierarchical processes (Botvinick & Plaut, 2004). CRU takes both sides. CRU is a hierarchical process because it is controlled by the outer loop. CRU also has hierarchical context representations that are operated on by a nonhierarchical retrieval process, based on matching (Figure 2C, 2D). The representation is hierarchical because the word command is set in the context vector for every keystroke in the word. This common element increases similarity of contexts within the word and decreases similarity of contexts from other words, which reduces intrusions and keeps CRU on track (see Figure 7). Other models structure hierarchical representations similarly, using common elements to connect separate representations (Botvinick & Plaut, 2004; Farrell, 2012; Vousden, Brown, & Harley, 2000). In CRU, the structure is in the similarities, not in associations. Processing is driven by information.

Plans, Programs, and Chunks

In the motor control literature, sequence production is often thought to result from a plan or a program or a chunk that specifies the sequence in advance and controls each of its steps (Diedrichsen & Kornysheva, 2015; Keele, 1968; Keele, Ivry, Mayr, Hazeltine, & Heuer, 2003; Rosenbaum, Inhoff, & Gordon, 1984; Schmidt, 1975). CRU explains skilled sequence production without any of these concepts. Sequences are not planned in advance. They are retrieved on the fly. Sequences are not programmed. They are driven by the contexts they create through their own actions. Sequences are not bound with strong associations like chunks (Estes, 1972; Yamaguchi & Logan, 2016). There are no associations between contexts and no associations between key locations. There are only associations between contexts and key locations. The structure that drives the sequence and makes it coherent is in the similarities among the traces, not in associations or preplanned steps. Plans, programs, and chunks may be important in other motor acts, but they do not seem to be necessary in skilled typing.

Plans may be important early in practice, when people are just acquiring skill and do not yet have the knowledge to support context retrieval. Programs may be important in crystallizing procedural knowledge at intermediate stages of practice, but they may be supplanted later on by the looser and more flexible process of context retrieval (Anderson, 1982; Fitts & Posner, 1967). Context retrieval may only emerge after years of practice (but see Logan, 1988).

Serial Order

Chaining and position coding. The problem of serial order has challenged theorists since the beginning of experimental psychology. The dominant theoretical traditions were staked out early. Ebbinghaus (1885) proposed an *associative chaining* mechanism, in which items are associated with succeeding (and preceding) items. Serial order is driven by the items that are retrieved: each retrieved item is the cue for the next. Ladd and Woodworth (1911) proposed a *position coding* mechanism, in which items are associated with position codes or context representations but not with each other. Serial order is driven by stepping through position codes: the item associated with each code is retrieved, but the item is not the cue for the next retrieval. Chaining and position coding were debated extensively in studies of maze learning (Hull, 1932, 1934; Tolman, 1948) and serial learning (Ebenholtz, 1963; Young, 1962).

The essential distinction between chaining and position coding lies in their assumptions about associations. Chaining assumes associations between items. Position coding assumes associations between items and position codes or items and contexts. CRU contains elements of both. Its associations between key locations and stored contexts are like position coding. However, its contexts are made of superimposed items (retrieved motor commands), so its associations between items and contexts may also be thought of as associations between items, like chaining (also see Howard & Kahana, 2002;

Lohnas et al., 2015; Polyn et al., 2009; more generally, see Botvinick & Plaut, 2006; Elman, 1990).

Lashley's influence. In 1951, Lashley published a landmark paper that shaped all subsequent work on serial order. He mounted strong arguments against chaining theories, claiming that they could not represent tasks like speaking, typing, or playing music, in which a small set of elements repeat in many different orders. He argued instead for hierarchical control, in which higher-level syntactic representations specify the order in which lower-level representations are executed. Many were convinced.

Lashley argued against behavioral chaining, in which the sensory consequences of one action were the stimulus for the next action (Hull, 1932, 1934), arguing that speaking, typing, and playing music were too fast for that kind of control. Interkeystroke intervals were shorter than sensory transmission times. Theorists overcame this objection by proposing chains of motor commands, in which the motor command for one action becomes the stimulus for the next (cf. von Holst & Mittelstaedt, 1950; Wolpert & Flanagan, 2001). So does CRU. The motor command for one keystroke becomes part of the context that retrieves the next keystroke. This removes the time for finger selection, movement, and processing sensory feedback from the loop, speeding up cycle time substantially.

The evolution of the current context in CRU is very much like behavioral chaining. It records the motor commands and their consequences. Just as a rat faces a different context after turning left instead of right, a typist (or CRU) faces a different context after typing F instead of G. I am surprised at Lashley's objection to this kind of chaining, as his "principal thesis . . . [was] that the input is never into a quiescent or static system, but always into a system which is already actively excited and organized" (p. 112). In CRU, the typist's own actions are an important part of that organized activity. We create the contexts we act in with our own actions.

Models of Serial Order

There are many models of serial order in studies of language, memory, and sequence learning. Rather than compare CRU to each one individually, I compare core features of the models, focusing on how they represent serial order, initiate retrieval, choose a response, and suppress retrieved responses.

Representation of serial order. Page and Norris (1998) proposed a model of serial recall with no associations to items or contexts. Instead, serial order is represented by the activation of items, with earlier items more active than later ones. Retrieval involves selecting the most active one and then suppressing it so the remaining items can compete. However, activation cannot represent repeated items, which occur often in typing, and activation has no long-term memory, which is required to support skills.

Theories of sequence learning generally endorse chaining (Abrahamse, Jiménez, Verwey, & Clegg, 2010; Helie, Roeder, Vucovich, Rünger, & Ashby, 2015; Keele et al., 2003). Participants typically learn one repeating sequence, and associations between items may be useful when a single sequence repeats. They are less useful when there are several sequences made of the same items (Lashley, 1951; see Figure 7).

Theories of speaking (Dell, 1986; MacKay, 1982; Vousden et al., 2000), typing (Rumelhart & Norman, 1982), playing music (Pfordresher, Palmer, & Jungers, 2007), and serial recall (Lewandowsky & Farrell, 2008) eschew chaining and endorse some form of position coding that may include hierarchical representation. Theories of speaking assume items are bound to syntactically ordered frames, which are independent of the items (Dell, 1986; Dell et al., 1997; MacKay, 1982; Vousden et al., 2000).

Many theories of serial recall assume that items are associated with contexts that are independent of the items. Henson et al. (1996) assumed contexts were defined by the distance from the beginning and end of the list. The same start and end markers are used for each list, which would make it difficult to represent large vocabularies. Estes (1955; Lee & Estes, 1981) assumed randomly drifting context vectors, so different contexts could occur with different lists. Burgess and Hitch (1999) assumed a time-varying context signal. Brown, Preece, and Hulme (2000) assumed banks of oscillators with different frequencies and defined context as the state of the oscillators when an item is presented.

Contexts that are independent of the items they are associated with may be good for remembering single presentations, but they are problematic for skills. The random contexts that occur on one expression of the skill are unlikely to occur on the next expression. The scale of contextual changes must change as performers acquire skill and speed up. The resonant frequencies of oscillators must change.

Howard and Kahana (2002; Lohnas et al., 2015; Polyn et al., 2009) developed models of serial order in free recall that rely on self-generated contexts, which provided the inspiration for CRU (also see Botvinick & Plaut, 2006; Elman, 1990). In their models, stored contexts are built of the items participants experience, updating according to Equation 1 with each new item. Retrieval occurs by feeding retrieved items back into the current context, changing the similarities between current and stored contexts. The same list appears in the same context each time it is presented, supporting skill acquisition, and different lists appear in different contexts, supporting large vocabularies of skilled action sequences.

It is surprising that models of serial recall have not embraced self-generated contexts. Many of them are built to model Baddeley's (1986) phonological loop (e.g., Burgess & Hitch, 1999), which represents items as motor commands for speaking. The current motor command must be chosen in the context of prior motor commands. That context might drive serial retrieval, as it does in CRU.

Initiating retrieval. CRU provides a new perspective on the thorny problem of how retrieval is instigated. Somehow the initial context has to be reestablished. It is not clear how this can happen when contexts are independent of the items. CRU is built on the idea that the initial context comes from the outer loop. It establishes a word command in the current context vector, and that instigates retrieval.

Choosing an item. Mechanisms for choosing an item are depicted in Figure 24. *Chaining* models choose the next item by activating it. Successive items are connected by positive associations, and activation flows along the chain. Chaining models have many problems (see above). *Successor inhibition* models (Bryden, 1967; Estes, 1972; Rumelhart & Norman, 1982) activate actions in parallel, but each action inhibits its successor, producing a gradient of activation in which earlier actions are more active than later ones. The item with the highest activation is chosen. These models have problems with repeated letters (see above). *Competitive queuing* models (Grossberg, 1978) activate items in parallel and make them compete in a winner-take-all network in which each item inhibits every other item. A filtering mechanism imposes a gradient of activation across the items that biases the competition so the item with the highest activation is chosen. Most theories of serial recall assume competitive queuing (see

Lewandowsky & Farrell, 2008) but it is often implemented as an approximation, using the Luce choice rule or softmax instead of dynamically evolving mutual inhibition. It is not clear the inhibition is necessary.

CRU assumes items are activated in parallel in proportion to their similarity to the current context (see Figure 5). The next item is chosen by a race among diffusions driven by the similarities. It is like a competitive queuing model with an independent race instead of mutual inhibition as the choice process. It is simpler than competitive queuing computationally. The finishing time distribution is given in Equation 5. The finishing time distribution for competitive queuing has to be estimated by simulation (Usher & McClelland, 2001). Theorists who are not committed to mutual inhibition in competitive queuing might consider a race process instead.

Response suppression. Successor inhibition and competitive queuing theories assume that the chosen item must be suppressed to remove it from the competition (also see Page & Norris, 1998). Otherwise, it would remain more highly activated than its competitors and repeat endlessly. Suppression is often implemented as self-inhibition, indicated by the red dotted lines in Figure 24. Most theories of serial order assume the chosen item is suppressed (Dell et al., 1997; Lewandowsky & Farrell, 2008; Rumelhart & Norman, 1982). CRU does not. The chosen item changes the current context, and that changes its similarities to the traces, so the chosen item is less likely to be repeated (see Figure 2). CRU resets all accumulators to zero after a choice is made (Logan & Gordon, 2001). This could be viewed as response suppression, but it is applied to all responses, not just the chosen one.

A recent study by Behmer and Crump (2017) suggests that there may be no response suppression in skilled typing. They had typists type continuous text and occasionally presented a probe letter to be typed immediately. They probed letters 1–3 positions before and after the next letter to be typed and found symmetrical priming for past and future letters (see Figure 25). Had there been response suppression, response times to probes that had just been typed (position −1) should have been much slower than response times to probes that were just about to be typed (position +1). These results support CRU's assumption that there is no response suppression.

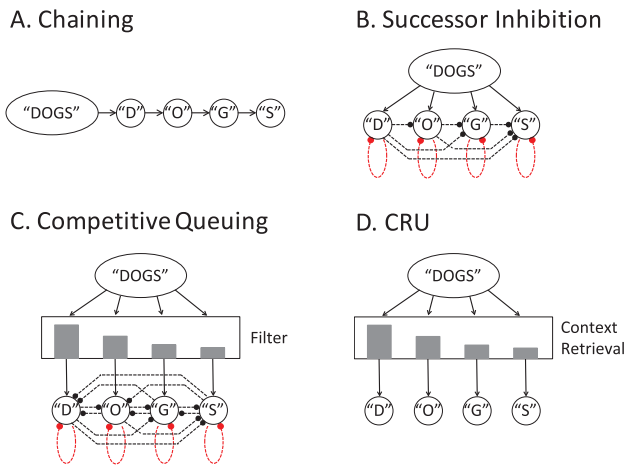

Figure 24. Chaining, successor inhibition, competitive queuing, and CRU. Solid lines with arrows represent activation. Broken lines with dots represent inhibition. Red broken lines with dots represent self-inhibition.

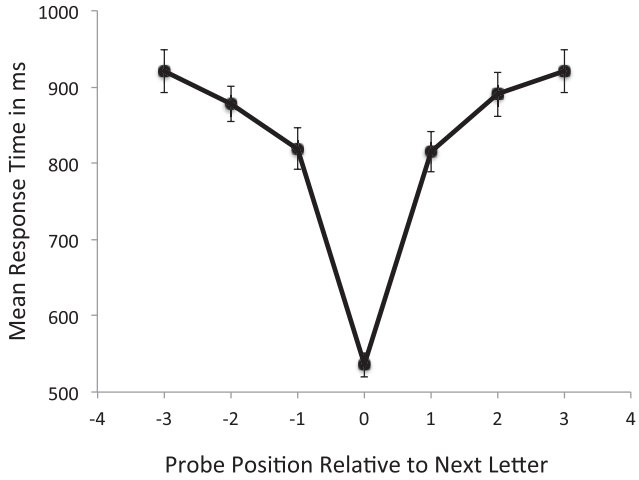

Figure 25. Mean response times to probe letters presented while typing continuous text, as a function of the position of the probe relative to the next letter to be typed. Negative values represent letters that had been typed. Positive values represent letters that have not yet been typed. The symmetry of the function around 0 suggests there is no suppression of the just-chosen letter. Data from Behmer and Crump (2017, Experiment 1). Error bars are standard errors of the mean.

Conclusion

Experts act without thinking because they recruit automatic control systems to take care of the details. I offered CRU as an example of an automatic control system that lets typists think in words while it chooses and sequences the required keystrokes. The key idea is that control is driven by a self-generated context that records the sequence of motor commands and the goal they were intended to achieve. We create the contexts for our actions. The simulations and model fits show that CRU is sufficient to explain automatic control. Future research comparing it to alternatives will be required to determine whether CRU is necessary as well as sufficient.

References

Abrahamse, E. L., Jiménez, L., Verwey, W. B., & Clegg, B. A. (2010). Representing serial action and perception. *Psychonomic Bulletin & Review, 17,* 603–623. http://dx.doi.org/10.3758/PBR.17.5.603

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review, 89,* 369–406. http://dx.doi.org/10.1037/0033-295X.89.4.369

Baddeley, A. D. (1986). *Working memory*. Oxford, UK: Clarendon Press.

Behmer, L. P., Jr., & Crump, M. J. C. (2017). The dynamic range of response set activation during action sequencing. *Journal of Experimental Psychology: Human Perception and Performance, 43,* 537–554. http://dx.doi.org/10.1037/xhp0000335

Botvinick, M., & Plaut, D. C. (2004). Doing without schema hierarchies: A recurrent connectionist approach to normal and impaired routine sequential action. *Psychological Review, 111,* 395–429. http://dx.doi.org/10.1037/0033-295X.111.2.395

Botvinick, M. M., & Plaut, D. C. (2006). Short-term memory for serial order: A recurrent neural network model. *Psychological Review, 113,* 201–233. http://dx.doi.org/10.1037/0033-295X.113.2.201

Boucher, L., Palmeri, T. J., Logan, G. D., & Schall, J. D. (2007). Inhibitory control in mind and brain: An interactive race model of countermanding saccades. *Psychological Review, 114,* 376–397. http://dx.doi.org/10.1037/0033-295X.114.2.376

Broadbent, D. E. (1982). Task combination and selective intake of information. *Acta Psychologica, 50,* 253–290. http://dx.doi.org/10.1016/0001-6918(82)90043-9

Brown, G. D. A., Preece, T., & Hulme, C. (2000). Oscillator-based memory for serial order. *Psychological Review, 107,* 127–181. http://dx.doi.org/10.1037/0033-295X.107.1.127

Bryden, M. P. (1967). A model for the sequential organization of behaviour. *Canadian Journal of Psychology/Revue canadienne de psychologie, 21,* 37–56. http://dx.doi.org/10.1037/h0082960

Burgess, N., & Hitch, G. J. (1999). Memory for serial order: A network model of the phonological loop and its timing. *Psychological Review, 106,* 551–581. http://dx.doi.org/10.1037/0033-295X.106.3.551

Cooper, R., & Shallice, T. (2000). Contention scheduling and the control of routine activities. *Cognitive Neuropsychology, 17,* 297–338. http://dx.doi.org/10.1080/026432900380427

Crump, M. J. C., & Logan, G. D. (2010a). Episodic contributions to sequential control: Learning from a typist's touch. *Journal of Experimental Psychology: Human Perception and Performance, 36,* 662–672. http://dx.doi.org/10.1037/a0018390

Crump, M. J. C., & Logan, G. D. (2010b). Hierarchical control and skilled typing: Evidence for word-level control over the execution of individual keystrokes. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 36,* 1369–1380. http://dx.doi.org/10.1037/a0020696

Crump, M. J. C., & Logan, G. D. (2010c). Warning: This keyboard will deconstruct—The role of the keyboard in skilled typewriting. *Psychonomic Bulletin & Review, 17,* 394–399. http://dx.doi.org/10.3758/PBR.17.3.394

Crump, M. J. C., & Logan, G. D. (2013). Prevention and correction in post-error performance: An ounce of prevention, a pound of cure. *Journal of Experimental Psychology: General, 142,* 692–709. http://dx.doi.org/10.1037/a0030014

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM, 7,* 171–176. http://dx.doi.org/10.1145/363958.363994

Dell, G. S. (1986). A spreading-activation theory of retrieval in sentence production. *Psychological Review, 93,* 283–321. http://dx.doi.org/10.1037/0033-295X.93.3.283

Dell, G. S., Burger, L. K., & Svec, W. R. (1997). Language production and serial order: A functional analysis and a model. *Psychological Review, 104,* 123–147. http://dx.doi.org/10.1037/0033-295X.104.1.123

Diedrichsen, J., & Kornysheva, K. (2015). Motor skill learning between selection and execution. *Trends in Cognitive Sciences, 19,* 227–233. http://dx.doi.org/10.1016/j.tics.2015.02.003

Ebbinghaus, H. (1885/1913). *On memory: A contribution to experimental psychology*. New York, NY: Teachers College, Columbia University.

Ebenholtz, S. M. (1963). Serial learning: Position learning and sequential associations. *Journal of Experimental Psychology, 66,* 353–362. http://dx.doi.org/10.1037/h0048320

Elman, G. (1990). Finding structure in time. *Cognitive Science, 14,* 179–211. http://dx.doi.org/10.1207/s15516709cog1402_1

Estes, W. K. (1955). Statistical theory of spontaneous recovery and regression. *Psychological Review, 62,* 145–154. http://dx.doi.org/10.1037/h0048509

Estes, W. K. (1972). An associative basis for coding and organization in memory. In A. W. Melton & E. Martin (Eds.), *Coding processes in human memory* (pp. 161–190). Washington, DC: Winston.

Farrell, S. (2012). Temporal clustering and sequencing in short-term memory and episodic memory. *Psychological Review, 119,* 223–271. http://dx.doi.org/10.1037/a0027371

Feit, A. M., Weir, D., & Oulasvirta, A. (2016). How we type: Movement strategies and performance in everyday typing. *CHI 2016: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Advance online publication. http://dx.doi.org/10.1145/2858036.2858233

Fendrick, P. (1937). Hierarchical skills in typewriting. *Journal of Educational Psychology, 28,* 609–620. http://dx.doi.org/10.1037/h0054049

Fitts, P. M., & Posner, M. I. (1967). *Human performance*. Oxford, UK: Brooks/Cole.

Flanders, M., & Soechting, J. F. (1992). Kinematics of typing: Parallel control of the two hands. *Journal of Neurophysiology, 67,* 1264–1274. http://dx.doi.org/10.1152/jn.1992.67.5.1264

Fodor, J. A. (1983). *Modularity of mind: An essay on faculty psychology*. Cambridge, MA: Bradford.

Grossberg, S. (1978). A theory of human memory: Self-organization and performance of sensory-motor codes, maps, and plans. In R. Rosen & F. Snell (Eds.), *Progress in theoretical biology* (Vol. 5, pp. 233–374). New York, NY: Academic Press. http://dx.doi.org/10.1016/B978-0-12-543105-7.50013-0

Grudin, J. (1983). Non-hierarchical specification of components in transcription typewriting. *Acta Psychologica, 54,* 249–262. http://dx.doi.org/10.1016/0001-6918(83)90038-0

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal, 29,* 147–160. http://dx.doi.org/10.1002/j.1538-7305.1950.tb00463.x

Heath, R. A., & Willcox, C. H. (1990). A stochastic model for interkeypress times in a typing task. *Acta Psychologica, 75,* 13–39. http://dx.doi.org/10.1016/0001-6918(90)90064-M

Helie, S., Roeder, J. L., Vucovich, L., Rünger, D., & Ashby, F. G. (2015). A neurocomputational model of automatic sequence production. *Journal of Cognitive Neuroscience, 27,* 1412–1426. http://dx.doi.org/10.1162/jocn_a_00794

Henry, F. M., & Rogers, D. E. (1960). Increased response latency for complicated movements and a "memory drum" theory of neuromotor reaction. *Research Quarterly of the American Association for Health, Physical Education, & Recreation, 31,* 448–458.

Henson, R. N. A., Norris, D. G., Page, M. P. A., & Baddeley, A. D. (1996). Unchained memory: Error patterns rule out chaining models of immediate serial recall. *Quarterly Journal of Experimental Psychology: Human Experimental Psychology, 49A,* 80–115. http://dx.doi.org/10.1080/713755612

Hershman, R. L., & Hillix, W. A. (1965). Data processing in typing: Typing rate as a function of kind of material and amount exposed. *Human Factors, 7,* 483–492. http://dx.doi.org/10.1177/001872086500700507

Howard, M. W., & Kahana, M. J. (2002). A distributed representation of temporal context. *Journal of Mathematical Psychology, 46,* 269–299. http://dx.doi.org/10.1006/jmps.2001.1388

Hull, C. L. (1932). The goal gradient hypothesis and maze learning. *Psychological Review, 39,* 25–43. http://dx.doi.org/10.1037/h0072640

Hull, C. L. (1934). The concept of the habit-family hierarchy and maze learning. *Psychological Review, 41,* 33–54. http://dx.doi.org/10.1037/h0070758

Hurlstone, M. J., Hitch, G. J., & Baddeley, A. D. (2014). Memory for serial order across domains: An overview of the literature and directions for future research. *Psychological Bulletin, 140,* 339–373. http://dx.doi.org/10.1037/a0034221

Jacoby, L. L. (1991). A process dissociation framework: Separating automatic from intentional uses of memory. *Journal of Memory and Language, 30,* 513–541. http://dx.doi.org/10.1016/0749-596X(91)90025-F

John, B. (1996). TYPIST: A theory of performance in skilled typing. *Human-Computer Interaction, 11,* 321–355. http://dx.doi.org/10.1207/s15327051hci1104_2

Kahneman, D. (2011). *Thinking, fast and slow*. New York, NY: Farrar, Straus & Giroux.

Keele, S. W. (1968). Movement control in skilled motor performance. *Psychological Bulletin, 70,* 387–403. http://dx.doi.org/10.1037/h0026739

Keele, S. W., Ivry, R., Mayr, U., Hazeltine, E., & Heuer, H. (2003). The cognitive and neural architecture of sequence representation. *Psycholog-*

ical Review, 110, 316–339. http://dx.doi.org/10.1037/0033-295X.110.2.316

Kragel, J. E., Morton, N. W., & Polyn, S. M. (2015). Neural activity in the medial temporal lobe reveals the fidelity of mental time travel. *The Journal of Neuroscience, 35,* 2914–2926. http://dx.doi.org/10.1523/JNEUROSCI.3378-14.2015

Kučera, H., & Francis, W. (1967). *Computational analysis of present-day American English*. Providence, RI: Brown University Press.

Ladd, G. T., & Woodworth, R. S. (1911). *Elements of physiological psychology: A treatise of the activities and nature of the mind from the physical and experimental point of view*. New York, NY: Charles Scribner's Sons.

Lashley, K. S. (1951). The problem of serial order. In L. A. Jeffress (Ed.), *Cerebral mechanisms in behavior* (pp. 112–136). New York, NY: Wiley.

Lee, C. L., & Estes, W. K. (1981). Item and order information in short-term memory: Evidence for multilevel perturbation processes. *Journal of Experimental Psychology: Human Learning and Memory, 7,* 149–169. http://dx.doi.org/10.1037/0278-7393.7.3.149

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics, Doklady, 10,* 707–710.

Lewandowsky, S., & Farrell, S. (2008). Short-term memory: New data and a model. In B. H. Ross (Ed.), *The psychology of learning and motivation* (Vol. 49, pp. 1–48). Burlington, MA: Academic Press. http://dx.doi.org/10.1016/S0079-7421(08)00001-7

Liu, X., Crump, M. J. C., & Logan, G. D. (2010). Do you know where your fingers have been? Explicit knowledge of the spatial layout of the keyboard in skilled typists. *Memory & Cognition, 38,* 474–484. http://dx.doi.org/10.3758/MC.38.4.474

Lo, C-C., & Wang, X.-J. (2006). Cortico-basal ganglia circuit mechanism for a decision threshold in reaction time tasks. *Nature Neuroscience, 9,* 956–963. http://dx.doi.org/10.1038/nn1722

Logan, F. A. (1999). Errors in copy typewriting. *Journal of Experimental Psychology: Human Perception and Performance, 25,* 1760–1773. http://dx.doi.org/10.1037/0096-1523.25.6.1760

Logan, G. D. (1982). On the ability to inhibit complex movements: A stop-signal study of typewriting. *Journal of Experimental Psychology: Human Perception and Performance, 8,* 778–792. http://dx.doi.org/10.1037/0096-1523.8.6.778

Logan, G. D. (1985). Skill and automaticity: Relations, implications, and future directions. *Canadian Journal of Psychology/Revue canadienne de psychologie, 39,* 367–386. http://dx.doi.org/10.1037/h0080066

Logan, G. D. (1988). Toward an instance theory of automatization. *Psychological Review, 95,* 492–527. http://dx.doi.org/10.1037/0033-295X.95.4.492

Logan, G. D. (1996). The CODE theory of visual attention: An integration of space-based and object-based attention. *Psychological Review, 103,* 603–649.

Logan, G. D. (2002). An instance theory of attention and memory. *Psychological Review, 109,* 376–400. http://dx.doi.org/10.1037/0033-295X.109.2.376

Logan, G. D. (2003). Simon-type effects: Chronometric evidence for keypress schemata in typewriting. *Journal of Experimental Psychology: Human Perception and Performance, 29,* 741–757. http://dx.doi.org/10.1037/0096-1523.29.4.741

Logan, G. D. (2017). Taking control of cognition: An instance perspective on acts of control. *American Psychologist, 72,* 875–884. http://dx.doi.org/10.1037/amp0000226

Logan, G. D., & Crump, M. J. C. (2009). The left hand doesn't know what the right hand is doing: The disruptive effects of attention to the hands in skilled typewriting. *Psychological Science, 20,* 1296–1300. http://dx.doi.org/10.1111/j.1467-9280.2009.02442.x

Logan, G. D., & Crump, M. J. C. (2010). Cognitive illusions of authorship reveal hierarchical error detection in skilled typists. *Science, 330,* 683–686. http://dx.doi.org/10.1126/science.1190483

Logan, G. D., & Crump, M. (2011). Hierarchical control of cognitive processes: The case for skilled typewriting. In B. Ross (Ed.), *The psychology of learning and motivation* (Vol. 54, pp. 1–27). Burlington, MA: Academic Press. http://dx.doi.org/10.1016/B978-0-12-385527-5.00001-2

Logan, G. D., & Gordon, R. D. (2001). Executive control of visual attention in dual-task situations. *Psychological Review, 108,* 393–434. http://dx.doi.org/10.1037/0033-295X.108.2.393

Logan, G. D., Miller, A. E., & Strayer, D. L. (2011). Electrophysiological evidence for parallel response selection in skilled typists. *Psychological Science, 22,* 54–56. http://dx.doi.org/10.1177/0956797610390382

Logan, G. D., Ulrich, J. E., & Lindsey, D. R. B. (2016). Different (key)strokes for different folks: How standard and nonstandard typists balance Fitts' law and Hick's law. *Journal of Experimental Psychology: Human Perception and Performance, 42,* 2084–2102. http://dx.doi.org/10.1037/xhp0000272

Logan, G. D., Van Zandt, T., Verbruggen, F., & Wagenmakers, E.-J. (2014). On the ability to inhibit thought and action: General and special theories of an act of control. *Psychological Review, 121,* 66–95. http://dx.doi.org/10.1037/a0035230

Logan, G. D., Yamaguchi, M., Schall, J. D., & Palmeri, T. J. (2015). Inhibitory control in mind and brain 2.0: Blocked-input models of saccadic countermanding. *Psychological Review, 122,* 115–147. http://dx.doi.org/10.1037/a0038893

Lohnas, L. J., Polyn, S. M., & Kahana, M. J. (2015). Expanding the scope of memory search: Modeling intralist and interlist effects in free recall. *Psychological Review, 122,* 337–363. http://dx.doi.org/10.1037/a0039036

MacKay, D. G. (1982). The problems of flexibility, fluency, and speed-accuracy trade-off in skilled behavior. *Psychological Review, 89,* 483–506. http://dx.doi.org/10.1037/0033-295X.89.5.483

MacNeilage, P. F. (1964). Typing errors as clues to serial ordering mechanisms in language behavior. *Language and Speech, 7,* 144–159. http://dx.doi.org/10.1177/002383096400700302

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. New York, NY: Henry Holt and Co.

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. http://dx.doi.org/10.1037/0033-295X.104.1.3

Miller, G. A., Galanter, E., & Pribram, K. H. (1960). *Plans and the structure of behavior*. New York, NY: Holt. http://dx.doi.org/10.1037/10039-000

Moors, A., & De Houwer, J. (2006). Automaticity: A theoretical and conceptual analysis. *Psychological Bulletin, 132,* 297–326. http://dx.doi.org/10.1037/0033-2909.132.2.297

Morton, N. W., & Polyn, S. M. (2016). A predictive framework for evaluating models of semantic organization in free recall. *Journal of Memory and Language, 86,* 119–140. http://dx.doi.org/10.1016/j.jml.2015.10.002

Norman, D. A. (1981). Categorization of action slips. *Psychological Review, 88,* 1–15. http://dx.doi.org/10.1037/0033-295X.88.1.1

Page, M. P. A., & Norris, D. (1998). The primacy model: A new model of immediate serial recall. *Psychological Review, 105,* 761–781. http://dx.doi.org/10.1037/0033-295X.105.4.761-781

Pashler, H., & Johnston, J. C. (1989). Chronometric evidence for central postponement in temporally overlapping tasks. *Quarterly Journal of Experimental Psychology: Human Experimental Psychology, 41,* 19–45. http://dx.doi.org/10.1080/14640748908402351

Pfordresher, P. Q., Palmer, C., & Jungers, M. K. (2007). Speed, accuracy, and serial order in sequence production. *Cognitive Science, 31,* 63–98. http://dx.doi.org/10.1080/03640210709336985

Polyn, S. M., Norman, K. A., & Kahana, M. J. (2009). A context maintenance and retrieval model of organizational processes in free recall. *Psychological Review, 116,* 129–156. http://dx.doi.org/10.1037/a0014420

Ratcliff, R., & Smith, P. L. (2004). A comparison of sequential sampling models for two-choice reaction time. *Psychological Review, 111,* 333–367. http://dx.doi.org/10.1037/0033-295X.111.2.333

Reason, J. (1990). *Human error*. New York, NY: Cambridge University Press. http://dx.doi.org/10.1017/CBO9781139062367

Reynolds, M., & Besner, D. (2006). Reading aloud is not automatic: Processing capacity is required to generate a phonological code from print. *Journal of Experimental Psychology: Human Perception and Performance, 32,* 1303–1323. http://dx.doi.org/10.1037/0096-1523.32.6.1303

Rieger, M. (2007). Letters as visual action-effects in skilled typing. *Acta Psychologica, 126,* 138–153. http://dx.doi.org/10.1016/j.actpsy.2006.11.006

Rieger, M., & Rieger, M. (2004). Automatic keypress activation in skilled typing. *Journal of Experimental Psychology: Human Perception and Performance, 30,* 555–565. http://dx.doi.org/10.1037/0096-1523.30.3.555

Rosenbaum, D. A., Inhoff, A. W., & Gordon, A. M. (1984). Choosing between movement sequences: A hierarchical editor model. *Journal of Experimental Psychology: General, 113,* 372–393. http://dx.doi.org/10.1037/0096-3445.113.3.372

Rosenbaum, D. A., Kenny, S. B., & Derr, M. A. (1983). Hierarchical control of rapid movement sequences. *Journal of Experimental Psychology: Human Perception and Performance, 9,* 86–102. http://dx.doi.org/10.1037/0096-1523.9.1.86

Rosenbaum, D. A., Loukopoulos, L. D., Meulenbroek, R. G., Vaughan, J., & Engelbrecht, S. E. (1995). Planning reaches by evaluating stored postures. *Psychological Review, 102,* 28–67. http://dx.doi.org/10.1037/0033-295X.102.1.28

Rosenbaum, D. A., Meulenbroek, R. J., Vaughan, J., & Jansen, C. (2001). Posture-based motion planning: Applications to grasping. *Psychological Review, 108,* 709–734. http://dx.doi.org/10.1037/0033-295X.108.4.709

Rumelhart, D. E., & Norman, D. A. (1982). Simulating a skilled typist: A study of skilled cognitive-motor performance. *Cognitive Science, 6,* 1–36. http://dx.doi.org/10.1207/s15516709cog0601_1

Salthouse, T. A. (1984). Effects of age and skill in typing. *Journal of Experimental Psychology: General, 113,* 345–371. http://dx.doi.org/10.1037/0096-3445.113.3.345

Salthouse, T. A. (1986). Perceptual, cognitive, and motoric aspects of transcription typing. *Psychological Bulletin, 99,* 303–319. http://dx.doi.org/10.1037/0033-2909.99.3.303

Salthouse, T. A., & Saults, J. S. (1987). Multiple spans in transcription typing. *Journal of Applied Psychology, 72,* 187–196. http://dx.doi.org/10.1037/0021-9010.72.2.187

Schauerte, B., & Fink, G. A. (2010). Focusing computational visual attention in multi-modal human-robot interaction. *Proceedings International Conference of Multimodal Interaction*. Advance online publication. http://dx.doi.org/10.1145/1891903.1891912

Schmidt, R. A. (1975). A schema theory of discrete motor skill learning. *Psychological Review, 82,* 225–260. http://dx.doi.org/10.1037/h0076770

Shaffer, L. H. (1975). Multiple attention in continuous verbal tasks. In P. M. A. Rabbitt & S. Dornic (Eds.), *Attention and performance V*. New York, NY: Academic Press.

Shaffer, L. H. (1976). Intention and performance. *Psychological Review, 83,* 375–393. http://dx.doi.org/10.1037/0033-295X.83.5.375

Shepard, R. N. (1987). Toward a universal law of generalization for psychological science. *Science, 237,* 1317–1323.

Shiffrin, R. M., & Schneider, W. (1977). Controlled and automatic human information processing: II. Perceptual learning, automatic attending, and a general theory. *Psychological Review, 84,* 127–190. http://dx.doi.org/10.1037/0033-295X.84.2.127

Snyder, K. M., Ashitaka, Y., Shimada, H., Ulrich, J. E., & Logan, G. D. (2014). What skilled typists don't know about the QWERTY keyboard. *Attention, Perception, & Psychophysics, 76,* 162–171. http://dx.doi.org/10.3758/s13414-013-0548-4

Snyder, K. M., & Logan, G. D. (2013). Monitoring-induced disruption in skilled typewriting. *Journal of Experimental Psychology: Human Perception and Performance, 39,* 1409–1420. http://dx.doi.org/10.1037/a0031007

Snyder, K. M., Logan, G. D., & Yamaguchi, M. (2015). Watch what you type: The role of visual feedback from the screen and hands in skilled typewriting. *Attention, Perception, & Psychophysics, 77,* 282–292. http://dx.doi.org/10.3758/s13414-014-0756-6

Soechting, J. F., & Flanders, M. (1992). Organization of sequential typing movements. *Journal of Neurophysiology, 67,* 1275–1290. http://dx.doi.org/10.1152/jn.1992.67.5.1275

Sternberg, S., Knoll, R. L., & Turock, D. L. (1990). Hierarchical control in the execution of action sequences: Tests of two invariance principles. In M. Jeannerod (Ed.), *Attention and performance XIII* (pp. 3–55). Hillsdale, NJ: Erlbaum.

Sternberg, S., Monsell, S., Knoll, R. L., & Wright, C. E. (1978). The latency and duration of rapid movement sequences: Comparisons of speech and typewriting. In G. E. Stelmach (Ed.), *Information processing in motor control and learning* (pp. 117–152). New York, NY: Academic Press. http://dx.doi.org/10.1016/B978-0-12-665960-3.50011-6

Teodorescu, A. R., & Usher, M. (2013). Disentangling decision models: From independence to competition. *Psychological Review, 120,* 1–38. http://dx.doi.org/10.1037/a0030776

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological Review, 55,* 189–208. http://dx.doi.org/10.1037/h0061626

Tzelgov, J. (1997). Specifying the relations between automaticity and consciousness: A theoretical note. *Consciousness and Cognition, 6,* 441–451. http://dx.doi.org/10.1006/ccog.1997.0303

Tzelgov, J. (1999). Automaticity and processing without awareness. *Psyche: An Interdisciplinary Journal of Research on Consciousness, 5,* 18–23.

Usher, M., & McClelland, J. L. (2001). The time course of perceptual choice: The leaky, competing accumulator model. *Psychological Review, 108,* 550–592. http://dx.doi.org/10.1037/0033-295X.108.3.550

Vandierendonck, A., Liefooghe, B., & Verbruggen, F. (2010). Task switching: Interplay of reconfiguration and interference control. *Psychological Bulletin, 136,* 601–626. http://dx.doi.org/10.1037/a0019791

Viviani, P., & Laissard, G. (1996). Motor templates in typing. *Journal of Experimental Psychology: Human Perception and Performance, 22,* 417–445. http://dx.doi.org/10.1037/0096-1523.22.2.417

von Holst, E., & Mittelstaedt, H. (1950). Das reafferenz princip: Wedlselwirkungen zwischen Zentrainervenststem und Peripherie [The principle of reafference: Interactions between the central nervous system and peripheral organs]. *Naturwissenschaften, 37,* 464–476. http://dx.doi.org/10.1007/BF00622503

Vousden, J. I., Brown, G. D. A., & Harley, T. A. (2000). Serial control of phonology in speech production: A hierarchical model. *Cognitive Psychology, 41,* 101–175. http://dx.doi.org/10.1006/cogp.2000.0739

White, C. N., Servant, M., & Logan, G. D. (2017). Testing the validity of conflict drift-diffusion models for use in estimating cognitive processes: A parameter-recovery study. *Psychonomic Bulletin & Review*. Advance online publication. http://dx.doi.org/10.3758/s13423-017-1271-2

Wolpert, D. M., & Flanagan, J. R. (2001). Motor prediction. *Current Biology, 11,* R729–R732. http://dx.doi.org/10.1016/S0960-9822(01)00432-8

Wu, C., & Lui, Y. (2008). Cuing network modeling of transcription typing. *ACM Transactions on Computer-Human Interaction, 15*. 1–45.

Yamaguchi, M., Crump, M. J. C., & Logan, G. D. (2013). Speed-accuracy trade-off in skilled typewriting: Decomposing the contributions of hierarchical control loops. *Journal of Experimental Psychology: Human Perception and Performance, 39,* 678–699. http://dx.doi.org/10.1037/a0030512

Yamaguchi, M., & Logan, G. D. (2014a). Pushing typists back on the learning curve: Contributions of multiple linguistic units in the acquisition of typing skill. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 40,* 1713–1732. http://dx.doi.org/10.1037/xlm0000026

Yamaguchi, M., & Logan, G. D. (2014b). Pushing typists back on the learning curve: Revealing chunking in skilled typewriting. *Journal of Experimental Psychology: Human Perception and Performance, 40,* 592–612. http://dx.doi.org/10.1037/a0033809

Yamaguchi, M., & Logan, G. D. (2016). Pushing typists back on the learning curve: Memory chunking in the hierarchical control of skilled typewriting. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 42,* 1919–1936. http://dx.doi.org/10.1037/xlm0000288

Yamaguchi, M., Logan, G. D., & Li, V. (2013). Multiple bottlenecks in hierarchical control of action sequences: What does "response selection" select in skilled typewriting? *Journal of Experimental Psychology: Human Perception and Performance, 39,* 1059–1084. http://dx.doi.org/10.1037/a0030431

Young, R. K. (1962). Tests of three hypotheses about the effective stimulus in serial learning. *Journal of Experimental Psychology, 63,* 307–313. http://dx.doi.org/10.1037/h0038534

Zbrodoff, N. J., & Logan, G. D. (1986). On the autonomy of mental processes: A case study of arithmetic. *Journal of Experimental Psychology: General, 115,* 118–130. http://dx.doi.org/10.1037/0096-3445.115.2.118

Appendix A

Error Corpus Method

Typists

We recruited 24 typists from the Vanderbilt community, 18 of whom identified as female, 5 as male, and 1 as gender fluid. Their mean age was 22.7 ($\pm SD$ = 3.3) years. Their mean typing experience was 13.6 ($\pm$3.4) years. All had formal training, averaging 20.3 ($\pm$12.9) weeks. Nineteen identified as standard typists. Five identified as nonstandard typists. Typing speed was no different for standard (82.7 $\pm$ 13.3 WPM) and nonstandard typists (78.9 $\pm$ 11.6 WPM), replicating Feit et al. (2016) and Logan et al. (2016).

Procedure

The procedure was approved by Vanderbilt University's Institutional Review Board, assuring compliance with ethical standards. Typists filled in consent forms, filled in a typing experience questionnaire, and indicated whether they used standard or nonstandard mapping, based on a picture of the hands and keyboard with the standard mapping coded in color. Then they were tested in individual rooms. The material to be typed was presented on a CRT attached to a PC. Responses were registered on a standard QWERTY keyboard. Typists sat about 60 cm from the screen. The paragraph to be typed appeared in 18 point font in the top portion of a 24.1 $\times$ 19.7 cm gray box centered on the screen. The keystrokes typists typed were echoed in the bottom portion of the box. The backspace key was disabled and typists were told not to correct their errors.

Typists first saw the paragraph presented on the screen, and then typed it as quickly and accurately as they could. When they finished, they clicked on a "Next" button on the bottom of their screen with a mouse, after which the screen cleared and the paragraph to be typed was presented again. Typists were allowed to take breaks between paragraphs. The computer did not begin timing until typists struck the first key. Words per minute was calculated by dividing the number of keystrokes by the time between the first and last keystroke to get keystrokes per minute, and then dividing by 5 to get words per minute. Keystrokes and timing were recorded. Errors were identified offline.

Border Collie Texts

Paragraph 5. It is difficult to know how man ever managed large flocks of sheep on the rough and hilly terrain of these areas without the help of these wonderful dogs. The strains that proved most adept at the specialized type of work required were highly prized and selectively bred from. This produced the sort of collie we know today. From looking at very old photographs, it is remarkable how little they have changed in the last hundred years or so. It proves that the early flockmasters knew well the type of dog that was built on the correct lines for the job it was intended to do.

Paragraph 6. One other sphere where border collies are most successful is in search and rescue. Dog handlers are required to go out and look for missing climbers and walkers. A lot of these people get lost in areas where sheep are grazed. Border collies have to range well ahead of the handlers in order to cover the maximum amount of ground, so they must be tested for their trustworthiness with sheep before training starts. It does show what an adaptable breed the border collie is, in that it can be taught to ignore an animal that it has been specifically bred to herd. Border collies are becoming more and more popular for this purpose.

(*Appendices continue*)

Paragraph 7. A border collie from the correct source can be a charming pet. However, dogs bred from a strong working line can become very frustrated and destructive if they find themselves in an environment where there is nothing for them to do. The job is the border collie's main reason for living. The desperate need to work is slightly diluted in certain lines of border collies that are bred for the show ring. It is important to remember that, although a border collie is usually quite happy to be a loving pet, he will need plenty of exercise, and preferably some occupation for his very able brain.

Paragraph 8. Sheep dog trials are a stylized form of farm work for the border collie. Most dogs that compete in trials do so in addition to their daily work on the farm. It is rare that a dog is kept exclusively for trials, as the border collie is too useful to be kept for a special occasion. Most of the exercises that a working dog performs on the trials field are slight modifications of the jobs they perform every day of their working lives, and most sheep farmers would be lost without their dogs. Most trial winners more than earn their keep when used on the daily farm work.

Appendix B

Fitting the Model

The model was fit to the data by calculating the probability that each keystroke was struck by multiplying the probabilities of context retrieval and finger selection (Kragel et al., 2015; Morton & Polyn, 2016). The probability of context retrieval was calculated in several steps beginning with the correct word and the erroneous version of it. A set of stored $N + 1$ context vectors was generated for the N letters in the correct word, and a set of $M + 1$ context vectors was generated for the M letters in the erroneous word, where M need not equal N. Each vector was made of 1032 elements, most of which were set to zero. The first 32 represented the letters and the space bar, defined by subtracting 96 from their ASCII codes. One element corresponded to each letter. The remaining 1000 represented words. Each word was represented by setting a single element to 1 and the rest to 0, so different words were represented by orthogonal vectors. The simulation that produced the predicted values held up to 500 different context vectors.

The fitting routine stepped through the letters in the erroneous word, calculating the probability of choosing that letter given the current context in effect at the time of retrieval. The probability was calculated by calculating similarities (dot products) between the current context vector and all stored vectors, which were used as drift rates in Equation 5. The key that was struck became i in Equation 5 and the other keys became j. The probabilities were produced by numerically integrating Equation 5 from 0 to infinity (i.e., Equation 4). To illustrate, consider Figures 4 and 5. The typist intends to type DOG and has not typed any letters yet. The gradient shows the drift rates for each letter in DOG. If the typist typed D, then drift rate i in Equation 5 would be 1.0. However, if the typist typed O erroneously, then the drift rate i in Equation 5 would be .866. After integration, the probabilities are quite different.

The current context matches the stored contexts up to the first error. If the error was made in the context retrieval process (i.e., if the error was another letter from the same word), the error is incorporated in the updated context (e.g., if the typist Types O instead of D, then the motor context will contain O). If the error was not from the word, it was not incorporated in the updated context. The correct letter was included instead. I did this for two reasons. First, I assume the current context contains copies of motor commands for key locations, not records of the keys that were actually struck. Updating motor commands is faster than updating the actions that result from motor commands, and so is more efficient. Second, I tried incorporating finger choice errors into the current contexts and found that the model invariably typed the letter it should have typed instead of the error (e.g., typing DFOG after typing DF) as if the finger choice error added noise to the current context and ran it through context retrieval again. These errors are intrusions. Typists often make substitution errors (DFG for DOG) instead of intrusions. I wanted to give the model the capacity to make substitution errors.

The probability of selecting the finger associated with the key was computed by calculating the distance between the correct key and the typed key. If the key was correct, distance = 0 and $L_i = 1$ (see Equation 6). If the key was not correct but the error was another letter from the word, I assumed it was a context retrieval error, so the key that was struck was the key that was intended, so distance = 0 and $L_i = 1$. If the erroneous key was not from the word, I calculated distance between it and the correct key. If the distance was less than 4 cm, I calculated the drift rate as $L_i = \exp(-S \cdot d_{ij})$. If the distance was greater than 4 cm, I assigned the key a probability of 0.0000001 regardless of the parameter values (β and S), which removed its influence in the data fitting. I chose this distance criterion because about half of the errors that are not from the word come from adjacent positions, likely because of the spatial confusions predicted by the model, but the other half come from more remote positions and are caused by processes outside the model. People may misalign their fingers with the keyboard, for example. According to Equation 7, drift rate for finger retrieval v_{ik} is the product of L_i and the association strength of the finger A_{ik}. In these fits, I assumed every typist used standard mapping perfectly, so $A_{ik} = 1$ for the standard fingers and 0 for all other fingers.

(*Appendices continue*)

The error rates for words in the error corpora themselves overestimate the error rates for words in paragraph typing because the error corpora exclude the vast majority of the words that were typed correctly. Given one error per 5-letter word, error rate in the corpus would be about 20%, whereas error rate for the paragraphs averaged 7.29%. To compensate for this, I added likelihoods for typing the erroneous words correctly and added them to the likelihoods for the error strings in the model fitting. I estimated correct likelihoods by taking the maximum probability for the keystrokes in the error string (to exclude the error keystroke) and multiplied it by $N^{*}(pc^{1/5})/(1 - pc^{1/5})$, where N is the number of letters in the word and pc is the proportion of words typed correctly in the paragraph. This ratio gives the number of times the letters in word would have been typed correctly.

I started each fit with the same starting values ($\beta = .5$ and $S = .2$) because it would take too long to run many fits with different starting values (2 hours per fit). Explorations with the model suggested the best fitting parameters did not depend much on the starting values. In cases where the best fitting values were close to the starting values, I reran the fits with different starting values and they converged on the same best-fitting values.

Appendix C

Keystroke Timing

To separate systematic and random variance in keystroke timing, I calculated the mean and standard deviation of interkeystroke interval for each transition between successive keystrokes in each corpus. The means over typists are displayed in Tables C1 (mean interkeystroke interval), C2 (standard deviation), and C3 (number of observations per typist).

To assess the effect of word length on first and subsequent keystroke latencies, I calculated the mean interkeystroke interval for each position in 3- to 7-letter words for each typist. The means across typists are displayed in Figure 23.

Table C1
The Average Mean Interkeystroke Intervals (IKSIs) in ms Across Typists as a Function of Sequence

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A		166		169			156	148	116			138	131	114		133		112	129	105	120	151		253	108	277
B					155							147			142			168	147		147					
C	135		168		183			102	136		123	142			96					205	110					
D	146			173	185				138			129			125			196			162					
E	147		209	207	168	207			127			146	137	135	96	118	210	95	139	109		194	100	269	114	
F	166				155	163			140			117			113			186			114					
G					138			111						148				168	138							
H	106				113				111			180			124				126	116	157					
I	170		146		125	170	137					155	139	112	110			123	128	120		144				155
J															113											
K	210				116				183				220	97					120							
L	114			115	121				127		121	153			188	129			144	146	168	154			136	
M	126	145			127				142						128	164			159		211					
N	118			110	113		110		126				216	146	115				145	113		182				
O		154	134	144	108		140				140	196	112	112	149	142		126	99	125	96	143	114			
P	134				128			156				150			132	154		118		163	191				144	
Q																					135					
R	142		244	194	104	223	223		134		122	135	122	160	102	160		177	163	195	165				115	
S			255		126	195		117	130			114			111	136			173	134	133					
T	148				133			106	104			125			119			179	113	155			332		168	
U	118		129		152		124		101			160	189	186		145		102	118	119						
V					156				129																	
W	133				117			117	108						117											
X			235		258				151																	
Y					97							207				138										
Z					178																					

Note. The columns represent IKSIs for typing the letter at the head of the column. The rows represent the letter that was typed before that letter (e.g., column B row A contains the IKSI for typing B after A).

(*Appendices continue*)

Table C2
The Average Within-Typist Standard Deviations in ms as a Function of Sequence

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A		42		51			56	34	53			58	45	45		49		42	42	38	35	32		61	35	60
B					46							39			54			27	54		38					
C	36		31		45			28	55		36	61			37					46	34					
D	48			41	42				58			73			46			33			71					
E	50		43	47	33	75			44			63	58	48	38	62	56	36	53	31		35	33	76	43	
F	64				39	47			61			51			41			42			42					
G					41			34						59				25	44							
H	52				54				61			43			50				45	35	43					
I	85		47		46	90	45					40	43	37	59			51	54	50		52				68
J															29											
K	115				39				40				69	33					36							
L	52			39	54				48		22	34			44	16			74	51	51	88			46	
M	42	45			68				36						39	84			52		48					
N	50			54	45		46		33				49	19	34				69	41		66				
O		47	47	66	28		54				60	30	26	27	23	70		58	52	40	26	50	57			
P	70				60			46				32			41	15		50		66	74				79	
Q																					61					
R	47		60	50	44	58	49		52		57	74	45	50	44	64		47	48	48	64				46	
S			55		53	49		51	53			39			50	60			40	53	46					
T	61				43			47	35			48			53			42	31	17			104		69	
U	29		35		42		31		30			38	30	33		81		33	55	45						
V					54				63																	
W	32				33			34	40						49											
X			83		79				85																	
Y					26							100				39										
Z					52																					

Note. The columns represent IKSIs for typing the letter at the head of the column. The rows represent the letter that was typed before that letter (e.g., column B row A contains the standard deviation of IKSI for typing B after A).

Table C3
The Average Frequency With Which Each Typist Experienced Each Transition in Words Typed Correctly

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A		21		24			19	19	30			54	17	96		17		105	39	66	18	19		17	18	17
B					52							21			62			33	17		17					
C	35		17		24			18	21		33	16			69					36	17					
D	28			18	74				22			35			46			18			16					
E	37		34	102	43	18			46			33	42	25	16	49	18	189	89	24		18	17	24	18	
F	73				17	17			20			33			69			45			18					
G					45			27						18				26	24							
H	88				240				40			19			40				17	17	18					
I	66		16		48	16	17					35	34	108	31			35	26	26		27				24
J															18											
K	15				37				25				15	54					17							
L	27			24	52				92		18	72			35	16			74	26	16	17			46	
M	41	17			43				26						65	16			17		36					
N	19			72	34		66		36				15	17	24				15	27		15				
O		18	22	24	19		46				19	50	34	46	19	35		191	49	23	39	24	44			
P	17				52			26				17			19	18		46		24	23				18	
Q																					18					
R	45		24	75	172	27	18		42		48	18	72	19	61	10		28	46	28	24				24	
S			18		46	18		27	24			17			35	22			51	84	17					
T	43				43			275	27			16			27			49	18	19			15		24	
U	16		17		18		18		23			36	16	27		18		15	39	17						
V					72				48																	
W	28				24			31	22						45											
X			15		16				17																	
Y					18							17				37										
Z					21																					

Note. Frequencies from words typed incorrectly are not included. The rows represent the letter that was typed before that letter (e.g., column B row A contains the average frequency of typing B after A).

Received April 11, 2017
Revision received December 13, 2017
Accepted January 3, 2018 ■