Psychological Review
2010, Vol. 117, No. 3, 713–758

0033-295X/10/$12.00 DOI: 10.1037/a0019738

The Spatial Coding Model of Visual Word Identification

Colin J. Davis
Royal Holloway, University of London

Visual word identification requires readers to code the identity and order of the letters in a word and match this code against previously learned codes. Current models of this lexical matching process posit context-specific letter codes in which letter representations are tied to either specific serial positions or specific local contexts (e.g., letter clusters). The spatial coding model described here adopts a different approach to letter position coding and lexical matching based on context-independent letter representations. In this model, letter position is coded dynamically, with a scheme called spatial coding. Lexical matching is achieved via a method called superposition matching, in which input codes and learned codes are matched on the basis of the relative positions of their common letters. Simulations of the model illustrate its ability to explain a broad range of results from the masked form priming literature, as well as to capture benchmark findings from the unprimed lexical decision task.

Keywords: visual word recognition, models, spatial coding model, masked priming, orthographic input coding

The experimental and theoretical analysis of the processes involved in visual word identification has been a focus of cognitive science research in the last few decades (for reviews, see Carr & Pollatsek, 1985; Jacobs & Grainger, 1994; Rastle, 2007; Rayner, 1998; Taft, 1991). Word identification is an integral component of reading and of language comprehension more generally, and hence, understanding this process is critical for theories of language processing. Beyond that, however, the study of isolated visual word identification has attracted researchers because it provides a means of addressing fundamental cognitive questions pertaining to how information is stored and subsequently retrieved. For a variety of reasons, the domain of visual word identification is extremely well suited to studying issues related to pattern recognition. First, printed words (particularly in alphabetic languages) have many advantages as experimental stimuli, given that they are well-structured, discrete stimuli with attributes (such as frequency of occurrence, legibility, spelling–sound consistency, etc.) that are relatively easy to manipulate and control in experimental designs. Second, a variety of tasks have been developed with which to measure the time that it takes to identify a word, and this has led to a particularly rich set of empirical findings. Finally, printed words are highly familiar patterns with which the great majority of literate people demonstrate considerable expertise. Skilled readers are able to recognize familiar words rapidly (typically within about 250 ms, e.g., Pammer et al., 2004; Rayner & Pollatsek, 1987; Sereno & Rayner, 2003), in spite of the fact that they must distinguish these words from among a pool of tens of thousands of words that are composed of the same restricted alphabet of letters. To the reader this process appears effortless, but to the cognitive scientist it remains somewhat mysterious.

This research was supported by Economic and Social Research Council Grants RES-000-22-3354 and RES-000-22-2662.Thanks are due to Jeff Bowers and Steve Lupker, who provided helpful feedback on an earlier version of this article, and to Samantha McCormick, who assisted with the preparation of the article.

Correspondence concerning this article should be addressed to Colin J. Davis, Department of Psychology, Royal Holloway, University of London, Egham Hill, Egham, Surrey TW20 0EX, England. E-mail: c.davis@rhul.ac.uk

The Lexicalist Framework

In models of visual word identification, the goal of processing is often referred to as *lexical access* or lexical retrieval. In the present article, I describe the same state as the point of lexical identification. Such a state has been referred to as a "magic moment" at which the word has been recognized as familiar, even though its meaning has not yet been retrieved (e.g., Balota & Yap, 2006). Indeed, the point at which lexical identification occurs can be thought of as the gateway between visual perceptual processing and conceptual processing. In the E-Z reader model of eye movements during reading (e.g., Reichle, Pollatsek, Fisher, & Rayner, 1998), the completion of lexical identification may be viewed as the point at which attention is shifted from the current word to the next word. At a functional level of description, at least, this way of thinking about lexical identification implies an internal lexicon (or word level) containing unitized lexical forms. As Andrews (2006) notes, a lexicalist perspective of this sort need not entail assumptions about the nature of lexical knowledge—in particular, whether this knowledge is subserved by localist or distributed representations. Nevertheless, a localist account is the most straightforward means of implementing a lexicalist view (for discussion of theoretical arguments favoring localist over distributed representations, see Bowers, 2002; Bowers, Damian, & Davis, 2009; Davis, 1999; Page, 2000). According to such a localist account, lexical knowledge is underpinned by the existence of (and connections involving) nodes that code specific words. In the strongest version of such a localist account it may even be postulated that there are individual cells in the brain that code for specific words (e.g., an individual neuron that codes the word *cat*; Bowers, 2009); in support of such an account, recent evidence with functional mag-

netic resonance imaging rapid adaptation techniques provides evidence for highly selective neuronal tuning to whole words in the cortical region that has been labeled the *visual word form area* (Glezer, Jiang, & Riesenhuber, 2009).

There is an alternative to the lexicalist view. Some proponents of parallel-distributed processing models have rejected not only the notion of localist word representations but also the lexicalist view (e.g., Plaut, McClelland, Seidenberg, & Patterson, 1996; Seidenberg & McClelland, 1989) and have proposed models of ostensibly lexical tasks that include no lexicon. Debates about whether such models capture the central features of lexical processing (indeed, whether such models can even explain how readers are able to distinguish words from nonwords) are ongoing (e.g., Besner, Twilley, McCann, & Seergobin, 1990; Bowers & Davis, 2009; Coltheart, 2004; Dilkina, McClelland, & Plaut, 2008; Sibley, Kello, Plaut, & Elman, 2009) and will not be rehearsed here. There is no extant parallel-distributed processing model that can simulate the empirical results that form the critical database for the present investigation, and thus I do not consider such models further in this article.

Subprocesses in Visual Word Identification

Within a lexicalist framework, successful word identification appears to involve a number of basic processes (e.g., Forster, 1992; Jacobs & Grainger, 1994; Taft, 1991). First, it is necessary for the reader to encode the input stimulus by forming some representation of the sensory input signal. This representation needs to encode both the identity and the order of the letters in the input stimulus. Second, this input code must be matched against abstract long-term memory representations—lexical codes. Third, the best matching candidate must somehow be selected from among the tens of thousands of words in the reader's vocabulary. The present article considers each of these processes. The primary focus is on the first two processes, investigating how sensory input codes are matched against lexical codes and the nature of the input and lexical codes that are used in this process. The resulting match values then feed into a competitive selection process. All three of these processes are modeled herein in a series of simulations.

A Discrepancy Between Theory and Data

The last decade has seen a surge of interest in orthographic input coding and lexical matching, resulting in a large body of empirical data (e.g., Bowers, Davis, & Hanley, 2005a; Christianson, Johnson, & Rayner, 2005; Davis & Bowers, 2004, 2005, 2006; Davis & Lupker, 2010; Davis, Perea, & Acha, 2009; Davis & Taft, 2005; Duñabeitia, Perea, & Carreiras, 2008; Frankish & Barnes, 2008; Frankish & Turner, 2007; Grainger, Granier, Farioli, Van Assche, & van Heuven, 2006; Guerrera & Forster, 2008; Johnson, 2007; Johnson & Dunne, 20xx; Johnson, Perea, & Rayner, 2007; Kinoshita & Norris, 2008, 2009; Lupker & Davis, 2009; Perea & Carreiras, 2006a, 2006b; Perea & Lupker, 2003a, 2003b, 2004; Peressotti & Grainger, 1999; Rayner, White, Johnson, & Liversedge, 2006; Schoonbaert & Grainger, 2004; Van Assche & Grainger, 2006; Van der Haegen, Brysbaert, & Davis, 2009; Welvaert, Farioloi, & Grainger, 2008; White, Johnson, Liversedge, & Rayner, 2008). In the majority of these experiments, researchers have used the masked form priming paradigm (Forster, Davis, Schoknecht, & Carter, 1987) to investigate the perceptual similarity of pairs of letter strings that differ with respect to letter substitutions, transpositions, additions, and deletions; converging evidence has also been reported recently with the parafoveal preview technique (e.g., Johnson & Dunne, 20xx; Johnson, Perea, & Rayner, 2007). The resulting empirical database provides strong constraints on models of visual word recognition.

The literature includes a variety of computational models of visual word recognition, including the original interactive activation (IA) model (McClelland & Rumelhart, 1981), extensions of the IA model (Grainger & Jacobs, 1994, 1996), dual-route models (dual-route cascaded (DRC), connectionist dual-process (CDP), and CDP+; Coltheart, Rastle, Perry, Langdon, & Ziegler, 2001; C. Perry, Ziegler, & Zorzi, 2007; Zorzi, Houghton, & Butterworth, 1998), and parallel-distributed processing models (Harm & Seidenberg, 1999; Plaut et al., 1996; Seidenberg & McClelland, 1989). However, for all their successes, none of the above models is able to account for the results reported in the articles cited in the above paragraph. This discrepancy between theory and data points to fundamental problems in the standard approach to orthographic input coding and lexical matching.

In Davis (1999) and in subsequent articles, I have argued that these problems stem from the commitment of previous models to orthographic input coding schemes that are context-dependent (in the sense that they are either position- or context-specific) and that a satisfactory solution to these problems requires a context-independent coding scheme (see Bowers et al., 2009, for a recent discussion of the same issue in a different domain, i.e., serial order memory). I have also argued that lexical selection involves a competitive process and that this has important implications for the interpretation of experimental data (e.g., Bowers, Davis, & Hanley, 2005b; Davis, 2003; Davis & Lupker, 2006; Lupker & Davis, 2009). In the present article, I show how a context-independent model of orthographic input coding and lexical matching can be embedded within a competitive network model of lexical selection. The resulting model, which I will refer to as the *spatial coding model*, provides an excellent account of a large set of masked primed lexical decision findings pertaining to orthographic input coding, as well as explaining benchmark findings from the unprimed lexical decision task. Additionally, the model explains a considerable proportion of the variance at the item level in unprimed lexical decision.

How the Spatial Coding Model Is Related to the SOLAR and IA Models

Davis (1999) developed the context-independent orthographic input coding scheme within the framework of the self-organizing lexical acquisition and recognition (SOLAR) model. This model was developed with the goal of explaining how visual word recognition is achieved in realistic input environments, that is, environments that are complex and noisy and that change over time, thereby requiring the model to self-organize its internal representations. The SOLAR model is a competitive network model (e.g., Grossberg, 1976) and, therefore, part of the same class of models as the IA model. However, the features of the SOLAR model that enable it to self-organize result in a model that is considerably more complex than the IA model. These features include mechanisms governing the learning of excitatory and

inhibitory weights, a novel means of encoding word frequency (and a learning mechanism that modifies internal representations accordingly), and a mechanism for chunking identified inputs and resetting the component representations. Though interesting in their own right, these features are not critical to the phenomena modeled here (e.g., masked priming effects are not strongly influenced by online self-organization processes). The model that I develop in the present article draws on key aspects of the SOLAR model, notably the spatial coding scheme described in Davis (1999), the superposition matching algorithm subsequently developed in Davis (2001, 2004; see also Davis & Bowers, 2006), and the opponent processing model of lexical decision described in Davis (1999) but does not include the learning or chunking mechanisms of the SOLAR model; it also incorporates simpler assumptions with respect to frequency coding and lateral inhibitory connectivity. Thus, one way to think about the spatial coding model described here is as a (slightly simplified) stationary (i.e., non–self-organizing) version of the SOLAR model.

Another way to think about the spatial coding model I develop here is as an exercise in the *nested modeling* strategy (Jacobs & Grainger, 1994) that has guided the development of many computational models of visual word recognition in recent years (e.g., Coltheart, Curtis, Atkins, & Haller, 1993; Coltheart et al., 2001; Davis, 1999; 2003; Davis & Lupker, 2006; Grainger & Jacobs, 1994, 1996; C. Perry et al., 2007). These models have adopted a cumulative approach in which the best features of existing models are preserved in new models. In particular, each of the models listed above has incorporated a version of the IA model. This choice may have been related partly to the initial success of the original model in explaining data from the Reicher-Wheeler task (McClelland & Rumelhart, 1981; Reicher, 1969; Rumelhart & McClelland, 1982), but also no doubt reflects the fact that this model captured many of the essential features of the localist, lexicalist framework in a way that enabled detailed modeling of the temporal characteristics of lexical identification. Thus, the above-cited work has established that extensions of the IA model can explain not only Reicher-Wheeler data (e.g., Grainger & Jacobs, 1994) but also a broad range of other empirical results from the perceptual identification task, the unprimed lexical decision task, and the masked priming variant of the lexical decision task (Davis, 2003; Davis & Lupker, 2006; Grainger & Jacobs, 1996; Jacobs & Grainger, 1992; Lupker & Davis, 2009). Furthermore, the IA model has been used to provide the lexical route of dual-route models of reading aloud (Coltheart et al., 2001; C. Perry et al., 2007).

Although the nested modeling approach entails retaining the best features of previous models, features that are at odds with critical data should be replaced. To this end, the spatial coding model retains central assumptions of the IA model—localist letter and word representations, hierarchical processing, lateral inhibition, frequency-dependent resting activities—while modifying the IA model's orthographic input coding and lexical matching algorithm. In effect, then, the spatial coding model grafts the front end of the SOLAR model onto a standard IA model. Indeed, as is shown in the Appendix, given an appropriate parameter choice, the original McClelland and Rumelhart (1981) model can be specified as a special case of the present model (thus, although I do not consider Reicher-Wheeler data here, there is at least one parameterization of the model that accommodates the same set of findings as the original model). Although I do not attempt it here, it would be possible to use the spatial coding model as the lexical route of a dual-route model of word reading, following the approach of Coltheart et al. (2001) and C. Perry et al. (2007).

Overview of the Present Article

This article is arranged into two parts. The first part describes the model. I begin by describing the spatial coding scheme. What distinguishes this coding scheme from other schemes is its commitment to position and context-independent letter representations. This aspect of spatial coding, combined with its approach to coding letter position and identity uncertainty, underlies its ability to explain data that are problematic for other models. I then describe an algorithm (called superposition matching) for computing lexical matches based on spatial coding; I also discuss a possible neural implementation of superposition matching.

The set of equations describing spatial coding and superposition matching makes it possible to compute a match value representing orthographic similarity for any pair of letter strings. The relative ordering of match values for different forms of orthographic similarity relations is consistent with some general criteria that have been adduced from empirical data (Davis, 2006). However, to evaluate the model properly, it is necessary to derive predictions that are directly relevant to the dependent variables measured in experiments on orthographic input coding. To this end, I embed the spatial coding and superposition matching equations within a model of lexical selection and then explain how this model can simulate lexical decision. The resulting model is able to make predictions concerning primed and unprimed lexical decisions.

In the second part of the article, I demonstrate the application of the spatial coding model. In particular, I present a set of 20 simulations that model critical data from the masked form priming paradigm, examining the effect of letter replacements, transpositions, reversals, and displacements. The results demonstrate the broad array of findings that are explained by (and in several cases were predicted by) the spatial coding model. I also show that the model can explain various benchmark findings from the unprimed lexical decision task.

Part 1: Description of the Model

Spatial Coding

Davis (1999) introduced spatial orthographic coding as a means of encoding letter order that solves the alignment problem (i.e., that supports position-invariant identification) and captures the perceptual similarity of close anagrams. This general method of encoding order has its origins in Grossberg's (1978) use of spatial patterns of node activity to code temporal input sequences, and similar coding schemes have been used by Page (1994) in a model of melody perception and by Page and Norris (1998) in their primacy model of serial recall. The fundamental principle underlying spatial orthographic coding is that visual word identification is based on letter representations that are abstract (position- and context-independent) symbols. According to this idea, the abstract letter identities used for orthographic input coding are abstract in an even more extensive sense than has previously been proposed in standard models: In addition to abstracting away from visual form

(e.g., case, size, and color), these letter identities abstract away from positional and contextual factors. Essentially, they are mental symbols of the form proposed in Fodor's representational theory of mind (e.g., Fodor, 1975). Thus, according to spatial coding, the same letter *a* node can activate in response to the words *ape*, *cat*, *star*, or *opera*.

The relative order of the letters in a letter-string is encoded by the pattern of temporary values that are dynamically assigned (tagged) to these letters. Different letter orderings result in different spatial patterns (hence the term *spatial coding*; note that the word *spatial* does not refer to visuospatial coordinates). Some examples of spatial coding are shown in Figure 1. These examples show the pattern of values over the *o*, *p*, *s* and *t* letter nodes for four different words: *stop*, *post*, *opts*, and *pots*. The values assigned to letter nodes in these examples correspond to the serial positions of the corresponding letters in the stimulus, for example, the first letter is coded by a value of 1, the second letter is coded by a value of 2, and so on. This is the most straightforward version of spatial coding. In previous descriptions, I have sometimes assumed a primacy gradient rather than a recency gradient (i.e., the first letter is assigned the largest value, the second letter is assigned the next largest value, and so on). The two versions are mathematically equivalent when using the superposition matching algorithm: All that is critical is that the values are assigned so as to preserve the sequence in which the letters occurred in the input string.

Figure 1 illustrates how anagrams may be coded by exactly the same set of letter representations but by different relative patterns across these representations. For example, the spatial pattern used to code the word *stop* is quite different from that which is used to code the word *pots*, whereas *pots* and *post* are coded by quite similar patterns. Nevertheless, the fact that the same set of representations is used in each case is the critical difference between this approach and position- or context-specific coding schemes, which would code the word *stop* with an entirely different set of representations than those used to code its anagram *pots*.

One point that is important to note (and which has frequently been misunderstood) is that the gradient of values in an orthographic spatial code is purely a positional gradient—it is not a weighting gradient. That is, letter nodes that are assigned larger values are not given greater weight in the matching process than are nodes that are assigned smaller values. To use an analogy, the position of the notes following a treble clef indicates the pitch of those notes, not their loudness or duration. Thus, assigning a value of 1 to the node that codes the first letter of a stimulus and a value of 4 to the node that codes the last letter of a (four-letter) stimulus does not imply that the last letter is four times as important as the first letter: The values of the spatial code convey information about position only. This is not to say that all letters are in fact always given equal weighting during lexical matching but rather that coding differences in letter weighting requires a separate dimension, as described below.

Coding uncertainty regarding letter position. The perceptual coding of both letter position and letter identity is subject to a considerable degree of uncertainty, particularly in the earliest stages of word perception following the initial detection of the stimulus (e.g., Estes, Allmeyer, & Reder, 1976). Position uncertainty is a fundamental characteristic of the visual input to the lexical matching system, and any plausible model of orthographic coding needs to incorporate uncertainty in the signals output by letter nodes. For simplicity, the following discussion assumes that position uncertainty is restricted to the input code and that the learned code is error free. In spatial coding, letter position uncertainty is modeled by assuming that the position codes associated with letter signals are scaled Gaussian functions rather than point values. Thus, the model includes a parameter called σ, which reflects the degree of letter position uncertainty. Similar assump-

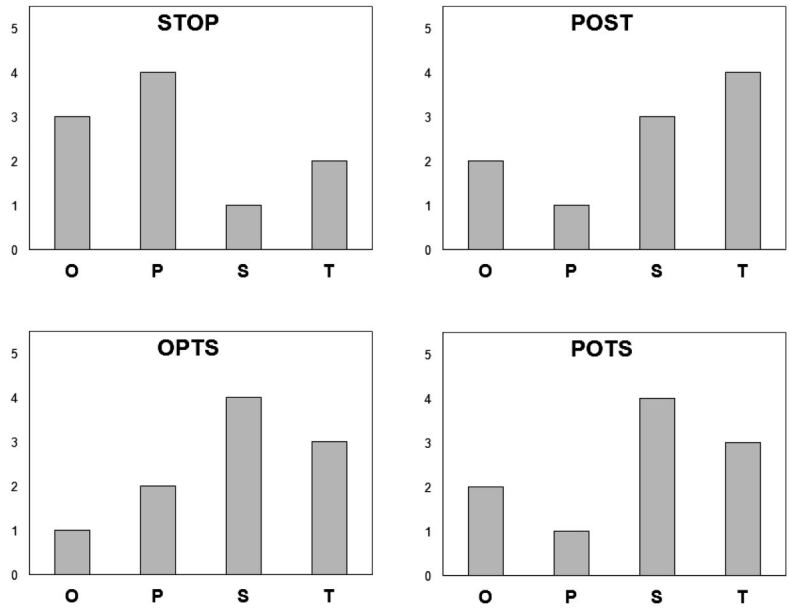

Figure 1. Examples of spatial coding. These examples show the pattern of values over the *o*, *p*, *s* and *t* letter nodes. The same letter nodes are used to code the words *stop*, *post*, *opts*, and *pots*, but with different dynamically assigned spatial patterns.

tions about the coding of letter position uncertainty have been made in other models of letter position coding (e.g., Gomez, Ratcliff, & Perea, 2008; Grainger et al., 2006). One way to depict this uncertainty is to plot the spatial code with error bars for each position code, as shown in Figure 2A. Another way to represent the spatial code is to rotate the axes so that the horizontal axis represents the position code, as shown in Figure 2B. The Gaussian-shaped uncertainty functions plotted in this figure are described mathematically by the equation

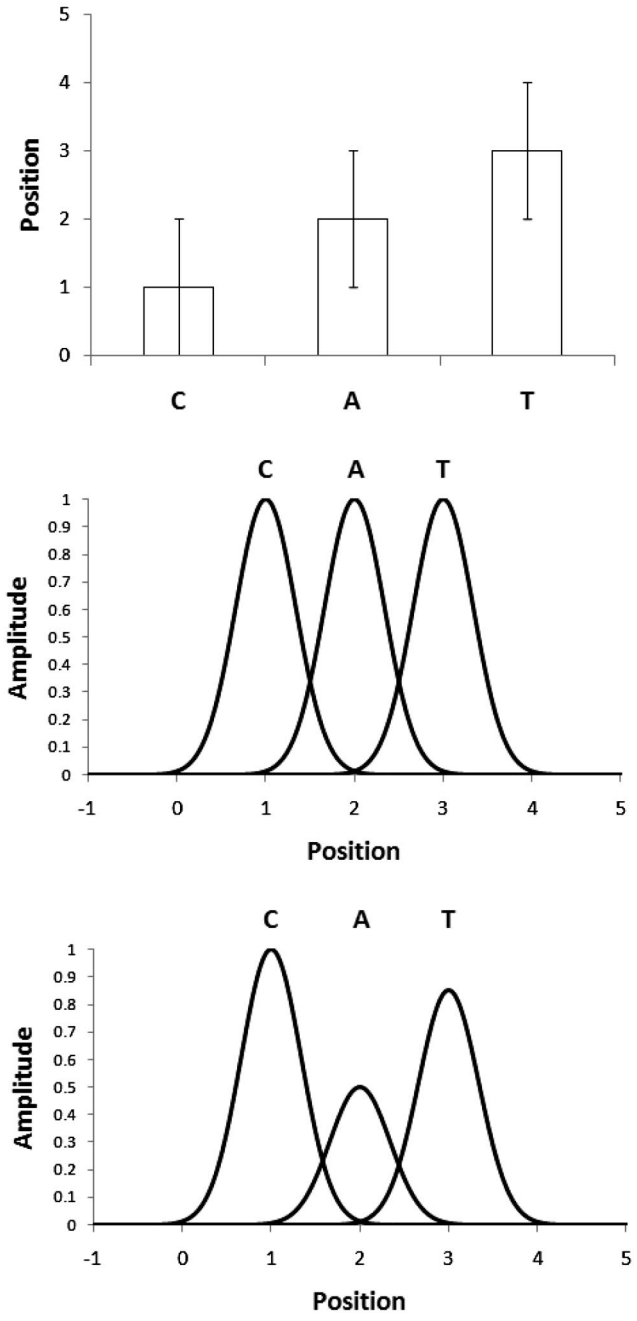

Figure 2. A: Spatial coding of *cat* with position uncertainty and error bars for each position code. B: Shows a different way of representing Figure 2A. C: Spatial coding of *cat* with position and identity uncertainty.

$$\text{spatial}_j(p) = e^{-\left(\frac{p-\text{pos}_j}{\sigma}\right)^2}, \quad (1)$$

where the subscript j indexes the letters within the spatial code and pos_j is the (veridical) serial position of the j^{th} letter within the input stimulus. For example, as *a* is the second letter of *cat*, the function coding the letter *a* in Figure 2B has the equation

$$\text{spatial}_A(p) = e^{-\left(\frac{p-2}{\sigma}\right)^2}. \quad (2)$$

Equation 2 holds wherever the word is fixated and whichever position-specific letter features are activated by the *a* in *cat*. At the same time, the specific value of 2 in this example is not critical—what is critical is the relative pattern among the letters within the spatial code. Thus, adding a constant to the values shown in the horizontal axis in Figure 2B would not disrupt the spatial code (e.g., values of 5, 6, and 7 for the letters *c*, *a*, and *t* would work equally well).

Factors affecting letter position uncertainty. A number of factors are likely to affect the magnitude of the σ parameter. One plausible assumption is that letter position uncertainty varies as a function of distance from fixation. That is, letters that are fixated are subject to relatively little position uncertainty, whereas letters in the parafovea may be associated with considerable position uncertainty. This relationship between letter position uncertainty and position of fixation provides the most likely explanation of the data of Van der Haegen et al. (2009), who observed that transposed letter (TL) priming effects increased considerably as the distance between the point of fixation and the TLs increased from zero to three letter widths. Davis, Brysbaert, Van der Haegen, and McCormick (2009) showed that the spatial coding model can fit these data well if σ is assumed to increase linearly as a function of distance from fixation.

Thus, the assumption that σ increases with distance from fixation helps to account for masked priming data; it is also supported by independent data from letter report tasks (Chung & Legge, 2009; Davis, McCormick, Van der Haegen, & Brysbaert, 2010). In general, however, this assumption is not useful for modeling data from the majority of published experiments, as fixation position is typically not controlled. However, another variable that is likely to affect σ is word length. Indeed, the assumption that σ increases with distance from fixation implies that the average value of σ for the letters in a word will tend to be larger for longer words than for shorter words, given that the letters in longer words will, on average, be further from fixation. This assumption is implemented in the simulations reported below by assuming the following linear relation between stimulus length and σ:

$$\sigma = \sigma_0 + \kappa_\sigma \text{stimulusLength}, \quad (3)$$

where σ_0 and κ_σ are parameters.

Coding uncertainty regarding letter identity. The spatial coding model also encodes uncertainty about letter identity. Letters for which there is considerable perceptual evidence in the input are coded by large letter activities, whereas letters that are only weakly supported by the perceptual input are coded by small letter activities. In the case in which there is no ambiguity concerning letter identity, each letter in the input stimulus is coded by a letter activity of 1.

The simultaneous coding of letter position and letter evidence necessitates a two-dimensional coding scheme. An example with this scheme is depicted in Figure 2C. Each letter node is associated with a two-dimensional function. The amplitude of the function represents the degree of letter evidence; in this example, it is assumed that there is less perceptual evidence supporting the middle letter than the two exterior letters.

$$\text{signal}_j(p, t) = \text{act}_j(t)\, e^{-\left(\frac{p-\text{pos}_j}{\sigma}\right)^2}. \quad (4)$$

As in Equation 1, the signal function in Equation 4 varies as a function of position, where the central tendency of the function represents the veridical letter position (pos_j), and the width of the function reflects the degree of letter position uncertainty (note that the label *spatial* in Equation 1 has been replaced by *signal* in Equation 4). The signal function in Equation 4 also varies over time (t). This reflects the fact that letter activity changes over time as initial letter ambiguity is resolved (the equation governing this change is described below). It would also be plausible to assume that position uncertainty varies over time (i.e., that uncertainty decreases with time), but for simplicity the present implementation assumes a fixed value of σ throughout time. The maximum value of the function in Equation 4 is 1, which occurs when the letter activity takes its maximum value of 1, $[\text{act}j(t) = 1]$ and $p = \text{pos}_j$.

The Gaussian-shaped functions assumed in the spatial coding model serve the same function as the Gaussian distributions in Gomez et al.'s (2008) overlap model. However, in the latter model, the setting of σ affects not only the horizontal extent of the position-uncertainty function but also the amplitude (height) of the function. This effect of σ is inconsistent with the two-dimensional coding scheme assumed here, in which the amplitude of the function represents the degree of letter identity uncertainty (i.e., it is important in the spatial coding scheme not to confound the coding of position uncertainty with the coding of letter identity uncertainty). This point is illustrated by Figure 2C, in which the amplitude of the letter *a* function is lower than that of the *c* and *t* functions (and the *t* function has a slightly lower amplitude than the *c* function) because this letter's identity is supported by weaker perceptual evidence, although its position is coded just as accurately (i.e., the three functions have equivalent horizontal extents). Another difference between the uncertainty functions in the two models is that the scaling of the Gaussian functions in the spatial coding model ensures that match values vary on a scale from 0 to 1.

Neural implementation of spatial coding. A neural instantiation of the two-dimensional spatial coding scheme was described by Davis (2001, 2004; see also Davis, in press). According to this account, the first dimension—the signal amplitude that is assumed to encode letter evidence—reflects the mean firing rate of a population of neurons that contribute to coding a given letter. The second dimension—the position code—reflects the phase with which the neurons within this population fire (with the σ parameter perhaps reflecting the noisy distribution of phase values). This *phase coding* hypothesis asserts that the position code is encoded in the phase structure of letter output signals. It is assumed that letter nodes output signals in a rhythmic fashion, such that these nodes "fire" with a fixed periodicity, for example, at times t, $t + P$, $t + 2P$, $t + 3P$, and so on, where p is a constant that represents the period length. Different letter nodes may fire at different times within this repeating cycle, in which case they are said to have different phases. The phase of the waves output by letter nodes is an index of relative position information: Earlier letters are coded by waves that are output earlier in the cycle. This is illustrated in Figure 3, which shows the letter signals output by the letter field when the input stimulus is the word *stop* (the right-hand side of the figure is described in the next section). In this case, waves are output by the letter nodes that code *s*, *t*, *o*, and *p* (in that sequence); the waves are shown at a point in time soon after the *p* letter node has output its signal. Note that the wave output by the *s* node is the most advanced at this point because it was output first, whereas the wave output by the *t* node is the second most advanced, and so on. As can be seen, there is some temporal overlap among these waves, reflecting letter position uncertainty.

Construction of the spatial (phase) code. Although a phase code could be constructed via a purely parallel process, the process I hypothesize here involves a very rapid serial process that scans from left to right across position-specific letter channels (in languages that are read from right to left, the scan would operate in that direction). This scan comprises a coding cycle that is divided into a sequence of phases, which correspond to the times within the cycle when a sequence coding mechanism (the spatial coder) sends rhythmic excitatory pulses to the letter level. This mechanism dynamically binds letter identity information with letter position information. I assume that this process ordinarily begins with an initial figure–ground segmentation process that determines the spatial extent of the stimulus and identifies the letter channels corresponding to the initial and final letters. The identification of the initial letter channel triggers the beginning of the coding cycle. The spatial coder sends an excitatory signal to that channel that causes active letter nodes within the channel to "fire," that is, to output signals to the word level. Because this is the start of the cycle, one can denote the resulting signals as having a phase of 1, although the absolute phase value is not critical. The spatial coder then moves its "attention" rightward to the next letter channel, so that its next rhythmic pulse causes letter nodes within that channel to fire with a phase of 2. This process continues until the spatial coder reaches the letter channel corresponding to the final letter. Thus, the spatial coder coordinates the letter output signals to the word level, causing active nodes within these channels to fire with a later phase for letters occurring later in the input stimulus. Davis (in press) discusses how a neural network architecture known as an

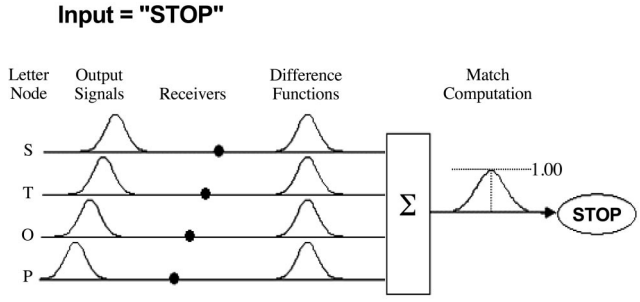

Figure 3. Schematic depiction of match computation at the *STOP* word node when the input stimulus is *stop*. Waves are output by the letter nodes that code *s*, *t*, *o*, and *p* (in that sequence). The waves are shown at a point in time soon after the *p* letter node has output its signal.

avalanche network (Grossberg, 1969) could implement the serial scan. The phase coding account provides a plausible description of how the theoretical ideas underlying spatial coding and superposition matching could be implemented within the brain (see Davis, in press, for further discussion of the neural plausibility of this implementation). Nevertheless, the success of spatial coding as a functional account does not depend on this particular neural instantiation being correct.

Superposition Matching

Superposition matching is a method for computing the match between two spatial codes: one that represents the current input to the system and another that represents the stored representation of a familiar word (the template). The template word is coded in the pattern of weights that connects the word node to the letter level, with the same spatial orthographic coding scheme that is used to code the input stimulus (e.g., a weight value of 1 for the first letter of the template, 2 for the second letter, and so on). The spatial coding model assumes that there is no uncertainty associated with the positions of the letters in the stored representation of familiar words, and hence, letter position is coded by point values rather than distributions. Lexical matching can thus be conceived of as an operation involving the comparison of two vectors: a signal vector representing the bottom-up input signals passed to the word node and a weight vector representing the template. As an example of the calculations involved in superposition matching, Table 1A illustrates the case in which the input stimulus is the word *brain* and the template is also the word *brain*. The first column of the table lists the letters of the template. The second column of the table lists the values of the spatial code for the input stimulus (i.e., the position-uncertainty functions are centered on these values). The third column of the table lists the values of the spatial code for the template. These values are identical to those in the first column because the stimulus is a perfect match to the template.

The superposition matching algorithm involves three steps. First, a signal-weight difference function is computed for each of the letters of the template. The central values of these functions are shown in the final column of Table 1A, and the signal-weight difference functions themselves are shown in Figure 4A. Signal-weight differences of 0 are computed for each of the comparison letters (this is always the case when the stimulus is identical to the template), and thus the signal-weight difference functions are perfectly aligned.

Table 1
Examples of Signal-Weight Difference Calculations Required for Superposition Matching

Input	Stimulus code	Template code	Difference
A. *brain*			
B	1	1	0
R	2	2	0
A	3	3	0
I	4	4	0
N	5	5	0
B. *wetbrain*			
B	4	1	3
R	5	2	3
A	6	3	3
I	7	4	3
N	8	5	3

The second step is to combine these signal-weight difference functions by computing a superposition function. The superposition of a set of signal-weight difference functions is simply the sum of the functions. The superposition function for the example I have been discussing is the top function in Figure 4A. Some examples of superposition functions for a variety of other cases are shown in Figure 4. For simplicity, these examples assume there is perfect letter identity information, that is, $\text{act}(t) = 1$.

The final step in the computation of the match value is to divide the peak of the superposition function by the number of letters in the template. In the example illustrated in Figure 4A, this division results in a match value of 1, which is the maximum match value.

A critical theoretical advantage of the superposition function is that it is sensitive to the relative values rather than the absolute values of the signal-weight differences. This is illustrated by the situation in which the input stimulus is a superset of the template, such as *wetbrain* (for the template *brain*). The signal-weight difference calculations for this stimulus are shown in Table 1B, and the resulting difference functions are depicted in Figure 4B. As can be seen, the five signal-weight difference functions are centered on 3 rather than on 0. Although the difference and superposition functions have been shifted by three positions (reflecting the fact that the letters of *brain* have been shifted three positions to the right in *wetbrain*), the superposition function has the same shape and peak, resulting in a match value of 1. This example illustrates how spatial coding, combined with superposition matching, supports position-invariant identification.

The examples depicted in Figure 4C–4F illustrate situations in which the input stimulus is (Figure 4C) an outer-overlap superset of the template, as in the case of *Brahmin* (for the template *brain*); (Figure 4D) a transposition neighbor of the template (e.g., the stimulus *Brian*); (Figure 4E) a nonadjacent transposition neighbor of the template (e.g., the stimulus *slate* for the template *stale*); or (Figure 4F) a backward anagram (e.g., the stimulus *lager* for the template *regal*). Note that the superposition function becomes broader and shallower (and consequently, the match value becomes smaller) across the latter three examples as the disruption to the relative positions of the letters increases. In particular, when the string is reversed, none of the signal-weight difference functions are aligned (see Figure 4F), and the match value is relatively small (.25).

Implementation of superposition matching. To implement superposition matching, I assume that the transmission of the spatial code to the word level goes via an intermediate set of nodes called *receivers*. For example, the *cat* word node is connected to separate receivers for the letters *c*, *a*, and *t*. These nodes compute signal-weight difference functions and output the result to the word node. Receiver nodes also serve the function of resolving the competition among the different outputs emanating from the letter level, as described below.

The phase coding hypothesis suggests that the connections between letter nodes and receiver nodes should be coded by a special kind of weight. Rather than a conventional weight, which multiplies the incoming input signal, these connections function as *delay lines*, which shift the phase of incoming input signals. This function is mathematically equivalent to the operation of comput-

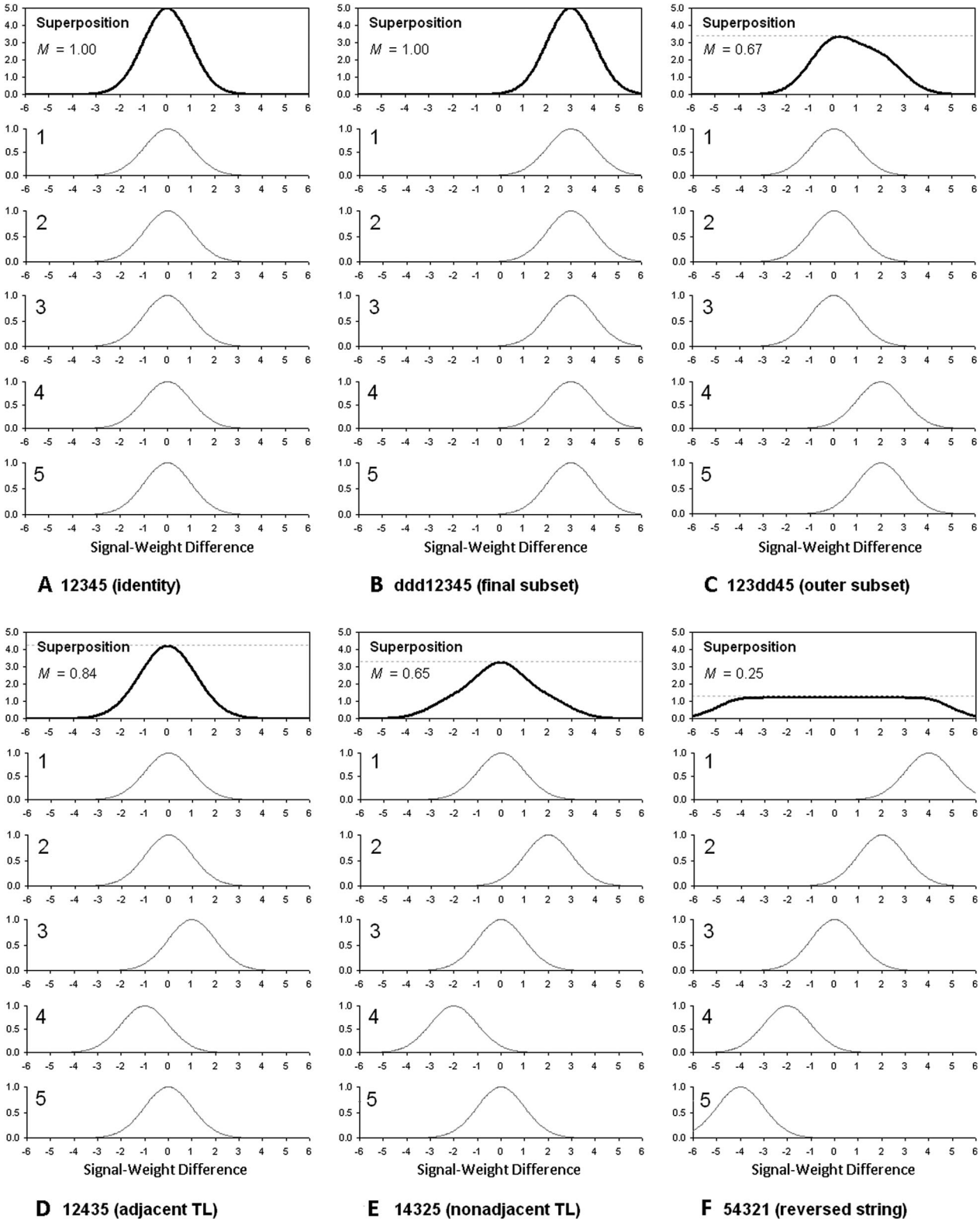

Figure 4. Examples of superposition matching. Figures A–F illustrate situations in which the input stimulus is (A) identical to the template word, (B) a final overlap superset of the template, (C) an outer-overlap superset of the template, (D) a transposition neighbor of the template, (E) a nonadjacent transposition neighbor of the template, or (F) a backward anagram. TL = transposed letter.

ing a signal-weight difference. The mathematical operation of superposition is realized by assuming that word nodes integrate the inputs coming from each of their receivers over relatively narrow temporal windows. In effect, word nodes act as temporal coincidence detectors. When there are few inputs to the node or when multiple inputs are out of phase with each other (as in the case of reversal anagrams like *lager–regal*), the summed input is relatively small, but when there are multiple inputs that are in phase (i.e., when they are temporally coincident, arriving at the word node at the same time), the summed input is relatively strong.

Formal description of match calculation. The following equations formalize the above description. I begin by considering a simplification, in which there is just one receiver node for each letter of the template, and this node receives input from just one letter node (below, I consider the more realistic case in which there are multiple receiver nodes for each letter of the template, which is required to handle repeated letters). Each of these receiver nodes is connected to the letter level by a delay line with value delay_{ri}, where the subscript i indicates that the receiver is attached to the i^{th} word node, and the subscript r is used to index the different receivers attached to this node (e.g., when the template is *cat*, the subscript r takes on values of 1, 2, or 3); in Equation 5 below, r is also used to index the letter node to which the receiver is attached. The value of delay_{ri} corresponds to the expected ordinal position of the corresponding letter within the template. (I note in passing that it would be possible to use complementary coding, in which the value of delay_{ri} is determined by subtracting the expected ordinal position of the letter from some fixed constant. The delay value would then be added rather than subtracted in Equation 5, which has a more ready physical interpretation. Nevertheless, exactly the same match values would result).

The receiver function is calculated by subtracting this delay value from the output signal of the letter node to which it is connected:

$$\text{receiver}_{ri}(p, t) = \text{spatial}_{r}(p, t) - \text{delay}_{ri}. \quad (5)$$

The superposition function is found by summing across the receiver functions for each of the template's receivers:

$$\text{superpos}_{i}(p, t) = \sum\nolimits_{r} \text{receiver}_{ri}(p, t). \quad (6)$$

The value of $\text{match}_{i}(t)$ is then

$$\text{match}_{i}(t) = \left(\frac{1}{\text{len}_{i}}\right) \text{superpos}_{i}[\text{resPhase}_{i}(t), t], \quad (7)$$

where len_{i} is the length of (i.e., number of letters in) the template, and $\text{resPhase}_{i}(t)$—the *resonating phase*—is defined as follows:

$$\text{resPhase}_{i}(t) = p^{*} \text{ such that } S_{i}(p^{*}, t) = \max[S_{i}(p, t)]. \quad (8)$$

That is, the resonating phase corresponds to the value of the signal-weight difference where the superposition function is at its peak; for example, for the situation depicted in Figure 4B, the resonating phase is 3. Basing $\text{match}_{i}(t)$ on the maximum instantaneous strength of the incoming superposition signal at time t implies that word nodes function as temporal coincidence detectors, as described earlier.

Dealing With Repeated Letters

A critical issue that must be addressed in the description of spatial coding is how to code stimuli that contain letter repetitions. Handling repeated letters requires that each letter should be coded by multiple letter nodes. To see why, consider the alternative whereby there is just a single letter node for each of the letters of the alphabet. In this scenario, coding any word that contained a repeated letter (e.g., *book*) would necessitate being able to simultaneously code the positions of two (or more) letters with a single letter node, which is not possible in a spatial coding scheme (as Davis, 1999, notes, attempting to do so would interfere with veridical coding of letter order).

Thus, rather than assuming a single receiver node for each letter of the template, it is necessary to assume there are multiple copies, or *clones*, of each receiver node. It is critical that the word node treats each of these different receivers as functionally equivalent; this is the principle of *clone equivalence*. That is, each receiver is equally capable of signaling to a word node the presence of a letter string that includes that letter. For example, the word node that codes *stop* activates in response to any set of *s*, *t*, *o*, and *p* receivers from which it receives temporally coincident (phase-aligned) signal functions.

The receiver nodes associated with a particular word node are organized into separate banks; that is, there is one bank of receiver nodes for each of the letters in the template. The present implementation assumes that there are position-specific letter channels (see Figure 6) and that each bank contains one receiver node for each letter channel, so that each of the nodes within a bank receives input from a corresponding letter node within a particular channel. For example, the *cat* word node is connected to three banks of receivers (for the letters *c*, *a*, and *t*, respectively), with the *a* bank containing one node that receives inputs from *a* in Channel 1, another node that receives inputs from *a* in Channel 2, and so on. I note in passing that it is also possible to implement receiver banks that have far fewer receivers within each bank (e.g., four is sufficient to code all English words).

The receiver function computed by an individual receiver within bank b of the i^{th} word node is calculated in the same way as before, but the notation includes an additional subscript:

$$\text{receiver}_{bci}(p, t) = \text{signal}_{cj}(p, t) - \text{delay}_{bi}. \quad (9)$$

The key difference between Equation 5 and Equation 9 is that the latter equation embodies the possibility that multiple receivers could activate for the same letter of the template. In particular, this situation arises when the stimulus includes one or more repeated letters.

Interactions Between Receiver Nodes

To deal with this situation appropriately, the model assumes that there are competitive-cooperative interactions between and within receiver banks. Specifically, there is winner-take-all competition between the receivers within each bank and between receivers in different banks that code separate occurrences of the same letter, and there are cooperative signals between receiver nodes that are in phase with each other (i.e., nodes that have computed equivalent signal-weight differences). There are also cooperative signals between receiver nodes that are in phase with each other, that is, nodes that have computed equivalent signal-weight differences. These competitive–cooperative interactions are weighted by letter

activity; that is, clones that receive strong letter signals carry greater weight than those that receive weak letter signals. The effect of these competitive-cooperative interactions is to select (at most) one winner within each bank (it is possible for a bank to contain no winners; for example, this occurs when the input stimulus does not contain the letter represented by that bank). One can define $\text{winningReceiver}_{bi}$ to denote the particular receiver that activates in bank b. Equation 6 is then modified to become

$$\text{superpos}_i(p, t) = \sum_b \text{winningReceiver}_{bi}(p, t). \quad (10)$$

When neither the stimulus nor the template contain repeated letters, it is straightforward to determine the winning receiver (it is the only receiver activated in the bank), and the situation is the same as described in Equations 5–8. The principle of clone equivalence implies that it does not matter which of the receivers in a bank activates for a given letter.

If the input stimulus has repeated letters, there will be at least one bank in which two or more receiver nodes become active. The identity of the winning receiver within this bank depends on the pattern of competitive and cooperative interactions between the full set of receivers. To illustrate, Figure 5A shows the signal-weight differences computed when the input stimulus is the word *stoop* and the template is also the word *stoop*. These differences are shown in a matrix, in which the columns of the matrix represent the five banks of receivers (corresponding to the five letters of the template) and the rows represent the different receivers within each bank, each of which receives input from a separate letter channel (only the first five receivers are depicted, as this is sufficient to show all of the critical functions). For the letters *s*, *t*, and *p*, the computations are straightforward. Only one letter clone in each bank receives a positive output, and the signal-weight difference is equal to 0 in each case; that is, these three letters occur in their expected position. For the remaining two comparison letters (the repeated letter *o*), there are two active receivers in each bank. That is, the first *o* in the stimulus *stoop* could represent the first or the second *o* in the template and likewise for the second *o* in the stimulus. For the observer, it is self-evident that the third letter in the stimulus corresponds to the third (rather than the fourth) letter of the template. The network determines this on the basis of the competitive–cooperative reactions among receivers. The presence of five receivers that compute a signal-weight difference of 0 results in this being the resonating phase (see Equation 8). As a consequence of cooperative signals between these phase-aligned receivers, the competition between *o* receivers is won by those nodes that share the resonating phase, that is, Clone 3 in the first *o* bank (Bank 3), and Clone 4 in the second *o* bank (Bank 4). The winning receivers are indicated in the figure by the differences shown in bold font. Here, the set of five equivalent signal-weight differences will result in a match value of 1, as is appropriate for a stimulus that perfectly matches the template.

The present approach avoids a problem with alternative methods of dealing with repeated items (e.g., Bradski, Carpenter, & Grossberg, 1994; Davis, 1999) that do not obey the principle of clone equivalence. Such methods do not explain how the embedded word *stop* can be identified in the stimulus *pitstop* because the *stop* node attends to the first occurrences of *p* and t in the stimulus and therefore sees the input as *p ts o* By contrast, the competitive–cooperative interactions among receivers described here ensure that it is the second *p* and t in *pitstop* that activate the *stop* template.

Another issue relating to how the model handles repeated letters arises when the template, and not the stimulus, contains repeated letters. An example of this situation is depicted in Figure 5B. Here, the template is again the word *stoop*, but the stimulus is the word *stop*. Although the stimulus contains only a single *o*, signal-weight differences are computed in both of the *o* receiver banks. The problems, then, are (a) how the network prevents the single oc-

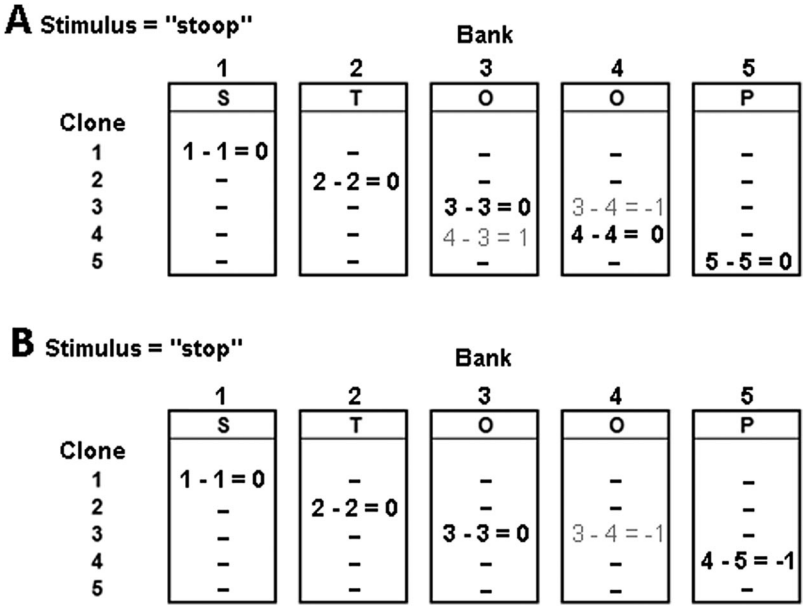

Figure 5. Illustration of computations performed by receiver nodes associated with the STOOP word node. A: Input stimulus = *stoop*. B: Input stimulus = *stop*.

currence of the letter *o* from doing double duty and contributing to both of the *o* receiver banks and (b) if it avoids the double-duty problem, how it chooses the correct receiver bank, so as to optimize the match value. These problems can be resolved by competition between receiver banks, which implements a one-letter, one-match rule that restricts stimulus letters from participating in more than one signal-weight match. The resonating phase for this set of signal-weight differences is 0 (there are three differences of 0 versus two differences of −1). Consequently, the receiver in the first *o* bank (Bank 3) attracts stronger cooperative signals than does the receiver in the second *o* bank (Bank 4), and this allows it to suppress the latter node. The assumption here is that there is winner-take-all competition not only between the receivers within each bank but also between receivers in different banks that receive inputs from the same letter node (e.g., Clone 3 in Bank 3 sends inhibition to Clone 3 in Bank 4 but not to Clone 4 in Bank 4). This competition between receivers prevents the single occurrence of the letter *o* from activating both *o* receiver banks. The four winning receivers are once again shown in bold, and the resulting signal weight differences (0, 0, 0, and −1) give rise to a match value of .72.

The present implementation of the model makes the simplifying assumption that the competitive–cooperative interactions between receivers occur instantaneously. In practice, however, a few cycles of processing may be required for within and between-bank competition to resolve potential ambiguities in the case of words with repeated letters. This additional processing time may explain the inhibitory effect of repeated letters on lexical decision latency reported by Schoonbaert and Grainger (2004).

Dynamic End-Letter Marking

The match calculations described thus far assign equal weight to all serial positions. However, there are various findings pointing to the special status of exterior letters, especially the initial letter. Transpositions that affect the exterior letters have a more disruptive effect on word identification than do transpositions of interior letters (e.g., Bruner & O'Dowd, 1958; Chambers, 1979; Holmes & Ng, 1993; Perea & Lupker, 2003a; Schoonbaert & Grainger, 2004; Rayner et al., 2006; White et al., 2008). Furthermore, participants are able to report the exterior letters of briefly presented letter strings with relatively high accuracy but make frequent location errors for interior letters (e.g., Averbach & Coriell, 1961; Merikle, Lowe, & Coltheart, 1971; Mewhort & Campbell, 1978).

Different models attempt to accommodate this aspect of orthographic input coding in different ways, that is, by assuming specialized end-letter nodes (Jacobs, Rey, Ziegler, & Grainger, 1998; Whitney, 2004), a smaller position-uncertainty parameter for the initial letter (Gomez et al., 2009), or specialized receptive fields for initial letter nodes (Tydgat & Grainger, 2009). The approach taken here shares similarities with each of the above mechanisms, as well as with recent models of serial recall (e.g., Farrell & Lelièvre, 2009).

Dynamic end-letter marking is an extension of the basic spatial coding model to accommodate the special status of exterior letters. Conceptually, this mechanism is straightforward: In addition to tagging each letter with a position code, the initial and final letters are explicitly marked as such; for example, the *s* and *p* in *stop* are tagged as the initial letter and the final letter, respectively. End-letter marking is envisaged as a process that complements spatial coding, providing an additional means of constraining the set of potential lexical candidates.

Exterior letter banks. End-letter marking is implemented in the spatial coding model via the assumption of specialized letter representations that explicitly (but temporarily) encode the exterior letters of the current stimulus. Thus, there is an initial letter bank that codes the initial stimulus letter and a final letter bank that codes the final stimulus letter (see Figure 6). Both of these banks contain one node for each letter of the alphabet (the figure shows only a subset of the nodes). There are excitatory connections between the two exterior letter banks and the word level; the weight of the connection from the j^{th} node within the initial letter bank to the i^{th} word node is denoted w_{ji}^{initial}, whereas the weight of the connection from the j^{th} node within the final letter bank to the i^{th} word node is denoted w_{ji}^{final}. It is assumed that these connections are pruned during the course of learning so that, ultimately, each word node has a positive connection to exactly one node in the initial letter bank and one node in the final letter bank.

Thus

$$w_{ji}^{\text{initial}} = \begin{cases} \dfrac{1}{\text{len}_i + 2} & \text{if template}_{i,1} = j \\ 0 & \text{otherwise} \end{cases}, \tag{11}$$

and

$$w_{ji}^{\text{final}} = \begin{cases} \dfrac{1}{\text{len}_i + 2} & \text{if template}_{i,\text{len}_i} = j \\ 0 & \text{otherwise} \end{cases}. \tag{12}$$

For example, Equation 11 implies that the weights from the initial letter bank to the *cat* word node are all 0 except for the connection from the *c* letter node in this bank. Likewise, Equation 12 implies

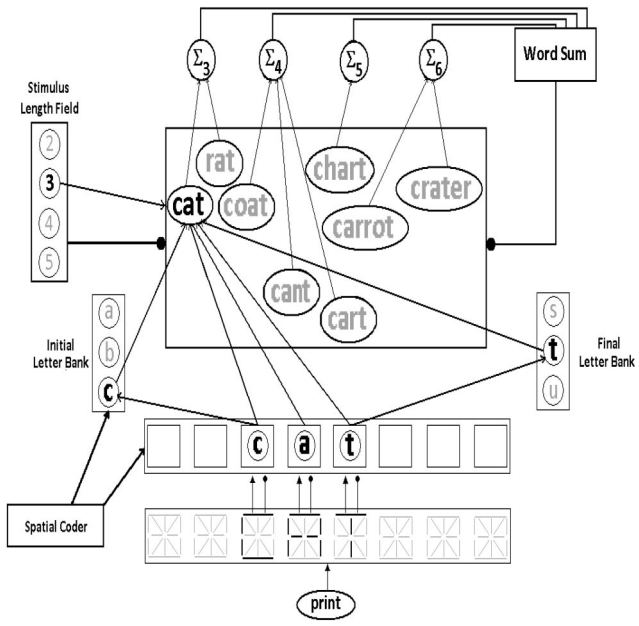

Figure 6. The spatial coding model. Figure depicts some of the nodes that are activated when the input stimulus is *cat*; only a subset of nodes and connections are shown.

that the weights from the final letter bank to the *cat* word node are all 0, except for the connection from the *t* node within this bank. The value of $\frac{1}{\text{len}_i + 2}$ for the positive weights reflects a simplifying assumption of weight normalization and weight equivalence (recall that len_i represents the length of the template). That is, the weights to the i^{th} node are normalized such that the incoming weights sum to 1 and so that all positive connections are of equivalent strength. The same assumption implies that the weight from receiver bank *b* to the i^{th}word node is

$$w_{bi} = \frac{1}{\text{len}_i + 2}. \quad (13)$$

For example, the *cat* word node receives five positive connections (two from the exterior letter banks and one each from the *c*, *a*, and *t* banks), and each of these connections has a weight of 1/5 = .2. The process by which these weights are learned is not modeled here, but this learning can be achieved quite readily with a Hebbian-type pattern learning algorithm (e.g., Grossberg, 1973). In alternative variants, the weights w_{bi} could vary across receiver banks, so that greater weights are assigned to letters that are more perceptually salient (e.g., the initial letter) or more informative with respect to lexical identity (e.g., consonants as opposed to vowels).

The activation of nodes within the exterior letter banks can be implemented as part of the function of the spatial coder. As noted above, word identification is assumed to begin with an initial figure–ground segmentation process that determines the spatial extent of the stimulus. When the letter channel corresponding to the initial letter is identified, a signal is sent to the initial letter bank, briefly opening a gate so that this bank can receive letter input signals. Likewise, when the letter channel corresponding to the final letter is identified, a signal is sent to the final letter bank, briefly opening a gate so that this bank can receive letter input signals. The upshot of this mechanism is that the initial letter bank temporarily mirrors the activity of the letter channel that corresponds to the initial letter of the current stimulus, and the final letter bank temporarily mirrors the activity of the letter channel corresponding to the final letter. Thus, the word identification system holds a temporary store of the initial and final letters of the stimulus from quite early in the identification process.

Incorporating exterior letter feedback in the match calculation. The incorporation of the signals from the exterior letter banks into the match calculation necessitates a slight modification to the previous equation. The revised equation is of the form

$$\text{match}_i(t) = \text{receiverOutput}_i(t) + \text{extLetterMatch}_i(t), \quad (14)$$

where

$$\text{receiverOutput}_i(t) = \sum_b w_{bi}\text{winningReceiver}_{bi}(\text{resPhase}_i, t), \quad (15)$$

and the weights w_{bi} are defined as in Equation 13. The exterior letter match is simply the dot product of the exterior bank letter activities with the corresponding weights to the word node:

$$\text{extLetterMatch}_i(t) = \sum_j w_{ji}^{\text{initial}}\text{act}_j^{\text{initial}}(t) + \sum_j w_{ji}^{\text{final}}\text{act}_j^{\text{final}}(t). \quad (16)$$

The inclusion of the normalized weights in Equations 15 and 16 ensures that the match values arising from Equation 14 are constrained to lie between 0 and 1 (and thus explicit division by len_i is unnecessary). Thus, Equations 3 through 16 define how the model assigns a spatial code and how it computes the match between spatial codes representing the stimulus and the template for a familiar word. These equations involve only two parameters, which determine how letter position uncertainty varies as a function of stimulus length (see Equation 3).

Evaluating the Match Values Produced by the Model

The set of equations presented above makes it possible to compute a match value representing orthographic similarity for any pair of letter strings. Table 2 lists match values for various types of orthographic similarity relationships, as computed by the spatial coding model with and without end-letter marking. Each example assumes a five-letter template word, though the input stimulus may contain fewer or more letters. As can be seen, the models with and without end-letter marking make quite similar predictions, but the addition of end-letter marking results in smaller match values for stimuli in which the end letters differ from the template and slightly larger values for stimuli with exterior letters that match those of the template.

The relative ordering of match values for the different forms of orthographic similarity relations shown in Table 2 is consistent with some general criteria that were proposed by Davis (2006), on a basis of a review of orthographic similarity data; for example, nearly adjacent transposition neighbors like *slate* and *stale* are more similar than double-substitution neighbors like *smile* and *stale*, but less similar than single-substitution neighbors like *scale* and *stale*). However, to properly evaluate the model it is necessary to derive predictions that are directly relevant to the dependent variables measured in experiments on orthographic input coding. To this end, I next describe how the spatial coding and superposition matching equations can be embedded within a model of lexical selection and how this model can simulate lexical decision.

Modeling Lexical Selection

Within the localist, lexicalist framework adopted here, lexical selection involves competition between lexical representations.

Table 2
Examples of Match Values for Spatial Coding Models With and Without End-Letter Marking

Type	Stimulus	Template	Without ELM	With ELM
Identity (12345)	*table*	TABLE	1.00	1.00
Initial superset (12345d)	*tablet*	TABLE	1.00	.86
Final superset (d12345)	*stable*	TABLE	1.00	.86
Outer superset (123d45)	*stable*	STALE	.83	.88
Adjacent TL (12435)	*trail*	TRIAL	.80	.86
Neighbor (d2345)	*teach*	BEACH	.80	.71
Neighbor (1d345)	*scale*	STALE	.80	.86
Neighbor once removed (13d45)	*sable*	STALE	.70	.79
Nonadjacent TL (14325)	*slate*	STALE	.62	.73
Double replacement (1dd45)	*smile*	STALE	.60	.71
Reversed (54321)	*lager*	REGAL	.22	.16

Note. ELM = end letter marking; TL = transposed letter.

Evidence supporting such lexical competition has been reported by Bowers, Davis, and Hanley (2005b) and Davis and Lupker (2006). The most well known model implementing this form of lexical selection is the IA model. As noted earlier, the spatial coding model retains many of the features of the IA model, including that model's localist letter and word representations, hierarchical processing, lateral inhibition, top-down feedback, and frequency-dependent resting activities. However, the orthographic input coding scheme and lexical matching algorithm of the original model are replaced by the spatial coding and superposition match algorithm described above.

Overview of Differences Between the Spatial Coding Model and the IA Model

The main differences between the spatial coding model and the original IA model are the input coding scheme and the way in which input stimuli are matched against word templates. However, there are also some other differences between the models that affect the present simulations. The original IA model was designed to handle words of a fixed length (four letters). When words of varying length are included in the vocabulary, there can be quite intense competition between subsets and supersets, for example, between words like *come* and *comet*. If the IA model's processes of lexical selection are not modified, it often fails to select the correct target word due to competition from subsets and/or supersets. As described below, the spatial coding model introduces two mechanisms to overcome this problem. There are also some differences between the models with respect to (a) the way word frequency influences word activation, (b) the nature of activity decay, (c) the way in which incompatible information in the stimulus inhibits word node activity, and (d) the nature of top-down feedback. As is shown below, the latter changes to the model have a small, positive impact on its ability to explain the data simulated in the second part of this article, although a good fit to the data can be obtained even without introducing these changes. That is, it is the input coding and matching assumptions that have been described already that are critical to explaining orthographic similarity data.

Architecture of the Model

The architecture of the spatial coding model is shown in Figure 6. The model is a localist neural network model: Each node within the model corresponds to a unique representation (e.g., a letter feature, a letter, or a word). As in the IA model, there are separate representational levels for letter features, letters, and words, and there are connections between nodes in adjacent levels. In addition, there are representational levels for coding exterior letters and for coding stimulus length. Nodes within the latter two levels receive inputs from the letter level and project connections to the word level. Furthermore, the model incorporates a spatial coding mechanism that coordinates the transmission of signals from the letter level to the word level.

The nodes within the feature and letter levels are divided into separate subsets representing different position-specific channels. Whereas the original IA model consisted of four channels, the present implementation includes 12. In other respects, these components of the model are equivalent to the original IA model. The representations at the letter level are treated as abstract letter identities, although in practice the Rumelhart & Siple (1974) font that is used to code letter features can only code uppercase letters. Although more plausible accounts of the features that readers use to identify letters are now available (e.g., Courrieu, Farioli, & Grainger, 2004; Fiset et al., 2008; Pelli, Burns, Farrell, & Moore-Page, 2006), McClelland and Rumelhart's (1981, p. 383) assumption that "the basic results do not depend on the font used" seems like a reasonable starting point.

Nodes at the word level are not position-specific. The only respect in which the word level in the spatial coding model differs from the IA model is the assumption of the intermediate receiver nodes that connect letter nodes to word nodes (these are not shown in Figure 6). As described above, the purpose of these nodes is to compute signal-weight difference functions, resolve the competition among the different outputs emanating from the letter level, and output the result to the word node.

As in the word level of the IA model, a crucial aspect of processing is that words compete with each other via lateral inhibition: This is the means by which the model selects the word node (or nodes) that best matches (match) the input stimulus. That is, the node that receives the greatest input from the letter level will dominate the activity at the word level and suppress the activity of competing word nodes. As shall be seen below, the presence of competitive interactions in the lexicon has important implications for the interpretation of the masked priming effects that have been the most common source of evidence in recent studies of letter position coding and lexical matching. As described below, the model implements lateral inhibition by means of the summation nodes shown at the top of Figure 6. This appears to be a neurally plausible method and is the most viable method of implementation from a modeling perspective (assuming direct lateral inhibitory connections between each pair of word nodes would require roughly 10^9 inhibitory connections for the current lexicon, versus approximately 30,000 in the present implementation).

Figure 6 also shows the exterior letter banks, which explicitly code the initial and final letters of the stimulus. Both of these banks contain one node for each letter of the alphabet (the figure shows only a subset of these nodes). There are excitatory connections between the two exterior letter banks and the word level (e.g., the *C* node in the initial letter bank sends an excitatory connection to the *CAT* word node, as seen in the figure).

Finally, the spatial coding model includes a stimulus length field, shown on the left-hand side of Figure 6 (again, the figure shows only a subset of the nodes within the field). The function of the nodes within this field is to explicitly code the length of the current input stimulus. Nodes of this type were previously proposed by Smith, Jordan, and Sharma (1991) to extend the IA model to processing words of varying length. As will be seen below, this assumption is not the only way to handle competition between words differing in length. Nevertheless, information about stimulus length presumably becomes available quite early in processing, based on both total letter level activity and independent visual input signals, and thus it seems plausible that this information is exploited by the visual word recognition system. Indeed, during normal reading, the visual system presumably exploits an estimate of the length of the next word to plan the saccade to that

word so that the eyes land close to the preferred viewing location (Rayner, 1979).

How Signals Flow Through the Model

Stimuli are presented to the model by setting the binary activities at the feature level. Active features then send excitatory signals to all of the letter nodes containing that feature and inhibitory signals to all of the letter nodes not containing that feature; these inputs result in the activation of letter nodes. The spatial coding mechanism then coordinates the output of letter signals to the word level, dynamically tagging these letter signals with a phase code that indicates relative letter position. These signals are intercepted by receiver nodes, which shift the phase of the signals (thereby implementing the previously described signal-weight difference computation) and resolve competition due to repeated letters. The signals output by receivers are then integrated at word nodes, which implement the superposition matching algorithm. Inputs from the exterior letter banks also contribute to the match value computed by word nodes. In addition to the match value, word nodes also compute a term that represents the mismatch between the input stimulus and the template. The net input to the word node is computed by combining these bottom-up match and mismatch signals with lateral inhibitory and excitatory signals, as well as length (mis)match signals from the stimulus length field. This net input drives a differential equation representing changes in activity over time. The other factors that influence this activity equation are exponential decay and a term that reflects the frequency of the word coded by the word node (thus high frequency words become activated more rapidly than low frequency words). When the stimulus is a word, the large match value computed by the node that codes that word will ensure that it soon starts to become more activated than do the others, and lateral inhibition within the word level then allows this word node to suppress its competitors. The time that it takes for the dominant word node to exceed the identification threshold is the critical factor affecting the speed of yes responses when the model simulates the lexical decision task. When the stimulus is not a word, the model will usually respond no, but the time that it takes to make this response will depend on the extent to which the stimulus activates nodes at the word level (i.e., very wordlike nonwords will take longer to reject than less wordlike nonwords).

Resting activities. Each node has a resting activity to which it decays in the absence of positive input, and this resting activity serves as the starting activity of the node at the beginning of each trial. The resting activity of letter nodes is assumed to be zero. The resting activity of word nodes was offset below zero as a function of log word frequency. The formula relating word frequency to word node resting activity is as follows:

$$\text{rest}_i = \text{FreqScale}\left(\frac{\log_{10}(\text{freq}_i) - \text{MaxF}}{\text{MaxF} - \text{MinF}}\right), \quad (17)$$

where MaxF represents the log frequency of the most frequent word in the model's lexicon (the word *the*) and MinF represents the log frequency of the most frequent word(s) in the model's lexicon. Equation 17 implies that the node coding the word *the* has a resting activity of zero and that nodes coding the least frequent words in the model's lexicon (those with frequencies of 0.34 per million words in the CELEX corpus, such as *behemoth*) have the lowest resting activity, determined by the parameter FreqScale. The latter parameter was set to .046 (i.e., the node coding *behemoth* has a resting activity of −.046), following the original IA model (see McClelland & Rumelhart, 1988).

Activation dynamics. The activation dynamics of letter and word nodes are governed by an activity equation that specifies how node activity should change on each cycle of processing. This activity equation is the same for letter and word nodes and takes the following form:

$$\text{act}_i(t + \Delta t) = \text{act}_i(t) + \text{shunt}_i(t)[\text{net}_i(t)] - \text{decay}_i(t) + \text{FreqBias}(\text{rest}_i). \quad (18)$$

This equation says that the instantaneous change in a node's activity depends on four factors: (a) the current activity (act_i), (b) the net input to the node (net_i), (c) the decay in node activity (decay_i), and (d) a bias input that favors higher frequency words. The current activity influences the instantaneous change in activity by moderating the effect of the net input, as can be seen in the following equation for shunt_i:

$$\text{shunt}_i(t) = \begin{cases} 1 - \text{act}_i(t) & \text{if net}_i(t) > 0 \\ \text{act}_i(t) - \text{ActMin} & \text{otherwise} \end{cases}. \quad (19)$$

The combination of Equations 18 and 19 implies that the effect of the net input decreases as the node activity approaches its maximum value (in the case of positive net input) or its minimum value (in the case of negative input). Positive inputs drive node activity toward a maximum of 1, whereas negative inputs drive node activity toward a minimum of ActMin; the parameter ActMin is set to −.2, as in the original IA model.

The third factor in Equation 18 represents exponential decay. This term is modified slightly from the original IA formulation so that node decay is match dependent. Nodes that match the current input stimulus well do not decay, whereas node activity decays rapidly for nodes that do not match the current stimulus well. For this purpose, the node's current match value, which varies between 0 and 1, is compared with a parameter called DecayCutoff. Thus,

$$\text{decay}_i(t) = 0, \quad (20a)$$

when $\text{match}_i(t) \geq \text{DecayCutoff}$, and

$$\text{decay}_i(t) = \text{DecayRate}[\text{act}_i(t)], \quad (20b)$$

when $\text{match}_i(t) < \text{DecayCutoff}$, where DecayRate is a parameter that controls the speed of the exponential decay in a node's activity. The computation of match values is described below.

The final factor in Equation 18, the $\text{FreqBias}(\text{rest}_i)$ term, is a negative input that effectively acts as a drag on the activation of low frequency words (recall that the maximum value of rest_i is 0) but has no effect on letter nodes (because all letter nodes are assumed to have zero resting activities). The introduction of distinct parameters for FreqBias and DecayRate differentiates the model from the IA model. When FreqBias is set equal to DecayRate and DecayCutoff is set to 1, Equation 20b always holds, and Equation 18 can be rewritten

$$\text{act}_i(t + \Delta t) = \text{act}_i(t) + \text{shunt}_i(\text{net}_i) - \text{DecayRate}[\text{act}_i(t) - \text{rest}_i], \quad (21)$$

which is identical to the original IA model. In the case where the net input is 0, the decay term in Equation 21 implies that node activity decays exponentially toward the node's resting activity, at a rate determined by DecayRate.

Computation of net input to letter nodes. Having explained the various components of the activity equation—its shunting term, exponential decay, and frequency bias—all that remains is to explain how the net input term is computed. In the case of letter nodes, there are two sources of input to the j^{th} letter node in channel c at time t:

$$\text{net}_{cj}(t) = \text{featureLetterInput}_{cj}(t) + \text{wordLetterInput}_{cj}(t). \quad (22)$$

The top-down wordLetterInput signal is similar to the IA formulation, but I delay detailed description of this component until the activation of word nodes by letter nodes has been described. The bottom-up featureLetterInput signal is computed in exactly the same way as in the original IA model, by taking the dot product of the feature activation vector and the feature-letter weight vector for that letter node; that is,

$$\text{featureLetterInput}_{cj}(t) = \sum\nolimits_k w_{kj}\text{feature}_{ck}(t), \quad (23)$$

where $\text{feature}_{ck}(t)$ is the binary activity of the k^{th} letter feature node in channel c at time t, and w_{kj} is the weight connecting that feature node to the j^{th} letter node. The value of this weight depends on the compatibility of the feature with the letter and the parameters α_{FL} and γ_{FL}, which represent the strength of feature-letter (FL) excitation and inhibition, respectively. Compatible features and letters (e.g., the feature representing the presence of a top horizontal bar and the letter *t*) are connected by an excitatory connection with strength $w_{kj} = \alpha_{FL}$, and incompatible features and letters are connected by an inhibitory connection with strength $w_{kj} = -\gamma_{FL}$.

Letter nodes can compute a match value by counting the proportion of positive feature signals they receive, or equivalently, via linear transformation of the featureLetterInput signal; that is,

$$\text{match}_{(cj)}(t) = \frac{\text{featureLetterInput}_{cj}(t) + 14\gamma_{FL}}{14(\alpha_{FL} + \gamma_{FL})}. \quad (24)$$

Equation 24 results in a match value that lies between 0 and 1 (the constant 14 reflects the number of letter features in the Rumelhart-Siple font). This match value can then be compared with the DecayCutoff parameter, as described in Equation 20.

Computation of net input to word nodes. The net input to the i^{th} word node can be decomposed into four sources, representing (a) the match between the input stimulus and the node's template, (b) a measure of the mismatch between the input stimulus and the node's template, (c) lateral inputs from within the word level, and (d) feedback from the stimulus length field (LW = letter-word):

$$\text{net}_i(t) = \alpha_{LW}[\text{match}_i(t)]^{\text{Power}} + \text{mismatch}_i(t) + \text{wordWord}_i(t) + \text{lenMismatch}_i(t). \quad (25)$$

In practice, word nodes should also receive feedback from other sources, such as phonological and semantic feedback. These inputs are not incorporated in the present implementation but could readily be added to the net input equation.

The computation of match_i—the first term in Equation 24—has already been explained. This match value is raised to a power (in order to contrast-enhance the input) and weighted by the parameter α_{LW}. I next describe how the remaining components of Equation 25 are computed.

Mismatch inhibition. The main source of bottom-up input to word nodes is the match value, which measures how well the current input stimulus matches the learned template. However, another (weak) source of bottom-up input to word nodes is a negative input that discounts evidence for a given word on the basis of stimulus letters that are incompatible with that word. This input helps to further constrain the set of potential lexical candidates, while avoiding problems associated with letter-word inhibition (e.g., Davis, 1999). The key difference between mismatch inhibition and the letter-word inhibition in the original IA model and related models (e.g., Coltheart et al., 2001; Grainger & Jacobs, 1996) is that mismatch inhibition takes account of the presence of mismatching letters but not the identity of these mismatching letters (and thus does not require any inhibitory letter-word connections). A word node is able to estimate the number of mismatching letters in the stimulus by subtracting a count of the number of letters that contribute toward the match with the template from the number of letters that are in the stimulus. The number of letters that contribute toward the match corresponds to the number of winning receivers, whereas total activity at the letter level (or activities at the stimulus length field) can be used to estimate the number of letters in the stimulus. In practice, the latter value is capped so that it does not exceed the number of letters in the template. Thus, the equation for computing mismatch inhibition is

$$\text{mismatch}_i = \gamma_{LW}[\min(\text{stimulusLength}, \text{len}_i) - C_i], \quad (26)$$

where C_i is the number of matching letters (i.e., the count of the positive signals from the receiver banks to the i^{th} word node) and γ_{LW} is a parameter weighing the influence of mismatch inhibition. The cap on the larger value in Equation 26 is to ensure that mismatch inhibition does not interfere with the recognition of familiar lexical constituents in complex words. For example, if the stimulus is *wildcat*, the mismatch is 3 (the number of letters in the template) minus 3 (the number of winning receivers) equals 0, rather than 7 (the number of letters in the stimulus) minus 3. In cases like this, the letters in *wild* are additional letters rather than mismatching letters, so it is appropriate to compute a 0 mismatch. Equation 26 also implies that mismatch inhibition cannot help to distinguish addition/deletion neighbors like *widow–window*, although it does help to distinguish substitution neighbors like *trail* and *trawl*. Furthermore, because the estimate of the number of letters that contribute toward the match is not dependent on position-specific coding, mismatch inhibition does not require that letters be in the "correct" position to avoid inhibiting a word node. For example, the *G* and *D* in the transposed-letter nonword *jugde* activate winning nodes at the receiver banks for the *judge* word node and thus do not count as mismatching letters. Note, however, that some anagrams will give rise to mismatch inhibition because the signal-weight difference functions for some constituent letters are so distant from the resonating phase. For example, assuming there is no extreme letter position uncertainty, the letters *e* and *j* in

eudgj do not activate winning nodes at the receiver banks for the *judge* word node, because they are too far from the resonating phase (which in this case is 0); thus, the asymptotic value of $\text{mismatch}_{\text{JUDGE}}$ is equal to 0 when the input stimulus is *judge* or *jugde* but is equal to 2 when the input stimulus is *eudgj*.

Lateral excitatory and inhibitory influences on word node activation. The wordWord_i component in Equation 25 has two components, one that is inhibitory, representing lateral inhibition at the word level, and one that is excitatory, representing the self-excitatory signal output by word nodes with positive activities:

$$\text{wordWord}_i(t) = -\gamma_{\text{ww}}\text{wordInhib}_i(t) + \alpha_{\text{ww}}\text{wordExcit}_i(t). \quad (27)$$

The relative contributions of these two components is weighted by the parameters $-\gamma_{\text{ww}}$ and α_{ww}.

Word–word inhibition. The wordInhib_i component in Equation 27 is computed in essentially the same way as in the IA model, in that it is calculated by summing across all of the positive word node activities (only active word nodes output a lateral inhibitory signal). The only difference is that lateral inhibitory signals in the spatial coding model are assumed to be length dependent. This assumption conforms to what Grossberg (1978) refers to as *masking field principles*. According to these principles, nodes that code longer words output stronger lateral inhibitory signals than nodes that code shorter words and are also assumed to dilute incoming lateral inhibitory inputs to a greater extent than nodes that code shorter words. These assumptions are implemented in the spatial coding model through a masking field weight that increases with the length of the template word. The masking field (mf) weight for the i^{th} word node is

$$\text{mf}_i = 1 + (\text{len}_i - 4)w_{\text{mf}}. \quad (28)$$

Equation 28 implies that the masking field weight equals 1 for words of four letters, which facilitates comparison with the original IA model. The parameter w_{mf} was set so that nodes that code seven-letter words output lateral inhibitory signals that are approximately twice as strong as those output by nodes that code four-letter words (e.g., $\text{mf}_{\text{PLANNER}} = 2.05$ versus $\text{mf}_{\text{PLAN}} = 1$).

Lateral inhibition is implemented by assuming the existence of a summation node that computes the total word inhibition signal. This approach avoids the need to assume specific inhibitory connections between each pair of word nodes. Figure 6 illustrates how this summation works for a subset of word nodes. Nodes that code words of different lengths output signals to different summation nodes, so that there are separate activity totals T_{len} for each different word length (len). For example, the T_3 summation node receives inputs from the *cat* and *rat* word nodes but not from nodes that code longer words such as *cart*, *chart*, or *carrot*. These signals are weighted by the masking field weight, so that longer words output a greater inhibitory signal. The total input to each of the length-dependent summation nodes can be written as follows:

$$T_{\text{Len}}(t) = \sum_{i\in\{\text{len}_i=\text{Len}\}}\text{mf}_i[\text{act}_i(t)]^+. \quad (29)$$

As can be seen in Figure 6, each length-dependent summation node sends a signal to a grand summation node. The total input to the latter node is

$$\text{wordSum}(t) = \sum_{\text{Len}} T_{\text{Len}}(t). \quad (30)$$

This value is then output by the grand summation node as an inhibitory signal to the word level. Following masking field principles, this inhibitory input is diluted at the word node according to the length of the template word. Thus,

$$\text{wordInhib}_i(t) = \frac{\text{wordSum}(t)}{\text{mf}_i}. \quad (31)$$

That is, an inhibitory input of a fixed magnitude has approximately twice as much impact on nodes that code four-letter words as on nodes that code seven-letter words.

Word–word excitation. The wordExcit_i component in Equation 27 represents the self-excitatory signal that a word node sends itself. Self-excitation is a common component of competitive networks, in which it can serve various adaptive functions (e.g., Carpenter & Grossberg, 1987; Davelaar, 2007; Grossberg, 1973; Wilson & Cowan, 1972). In the original IA formulation, self-excitation is included in the form of a term that ensures that word nodes do not inhibit themselves (i.e., a word node effectively subtracts its own activity from the incoming lateral inhibitory signal). Thus, the strength of the self-excitatory signal corresponds to the activity of the word node:

$$\text{wordExcit}_i(t) = [\text{act}_i(t)]^+, \quad (32)$$

where the $[\text{act}_i]^+$ notation indicates that only nodes with positive activities output a self-excitatory signal. The parameter α_{ww}, which weights self-excitatory signals, is set to be slightly larger than the parameter γ_{ww}, which weights lateral inhibitory signals, and thus self-excitation is used not only to cancel out self-inhibition but also to enhance the competitive process, enabling the best matching node to more rapidly suppress its competitors.

Length-matching. The lenMismatch_i term in Equation 25 represents the feedback signal from the stimulus length field, which consists of a net inhibitory signal to word nodes that do not match the length of current input stimulus. It is assumed that through a process of Hebbian learning, each word node develops a positive connection to the stimulus length node with ordinality corresponding to the length of the template, for example, the *cat* node will develop a connection to the stimLen_3 node. Thus, the weight v_{ni} from the n^{th} stimulus length node to the i^{th} word node is

$$v_{ni} = \begin{cases} 1 & \text{if } n = \text{len}_i \\ 0 & \text{otherwise} \end{cases}. \quad (33)$$

The facilitatory signal that the word node receives when the stimulus length matches the template length balances a nonspecific inhibitory signal that the stimulus length field sends to the word level; the strength of the latter signal corresponds to the total field activity (i.e., 1 when there is a letter string present and 0 otherwise). This gives

$$\text{lenMismatch}_i(t) = \gamma_{\text{len}}\left[\sum_n (\text{stimLen}_n(t) - \sum_n v_{ni}\,\text{stimLen}_n(t)\right]. \quad (34)$$

That is, if the length of the input stimulus does not correspond to its template length, the word node receives an extra inhibitory input that is weighted by the parameter γlen. A relatively small setting of this parameter is used in the current model, so that length mismatch inhibition does not prevent addition and deletion neighbors from becoming activated and interfering with identification of the target, as is required by the empirical evidence (e.g., Bowers et al., 2005a; Davis & Taft, 2005; Davis, Perea, & Acha, 2009).

The activation of nodes within the stimulus length field can be achieved via various sources of input. An approximate (but potentially unreliable) source of information regarding the length of the input stimulus is provided by visual signals with low spatial frequency. Activity at the letter level provides a somewhat more reliable source of information regarding the length of the input stimulus. At asymptote, only one letter is active within each channel, and this node will have the maximum letter node activity (of one), and thus the asymptotic total activity at the letter level is equivalent to the stimulus length. A further source of information regarding the likely length of the input stimulus is prior history; for example, it is common in masked priming experiments to use target stimuli of a fixed length. A full implementation of the stimulus length field would combine these various inputs and use lateral inhibitory interactions to select a single node. The present implementation takes the simpler approach of setting the stimulus length nodes directly, such that the activity of the node corresponding to the target stimulus length is set to one and the other stimulus length nodes are set to 0. For example, when the target stimulus is *cat*, the activity of the stimLen_3 node is set at one, whereas the activity of other stimulus length nodes is set at 0.

Top-down feedback from word nodes to letter nodes. As in the original IA model, top-down feedback is assumed to occur between the word and letter levels. However, the switch to position-independent letter coding necessitates a slightly different implementation of this top-down feedback. For example, if the stimulus *wildcat* leads to activation of the *cat* word node, this node should send feedback to the letter nodes that code positions five through seven rather than those for the first three positions. A word node can use the resonating phase to determine which letter channel should receive feedback signals. In particular, define

$$c_j = \text{IC} + \text{resPhase}_i + \text{pos}_j - 1, \tag{35}$$

where IC is the channel corresponding to the leftmost letter of the stimulus, resPhase_i is the resonating phase of the i^{th} word node (see Equation 8), and pos_j is the veridical position of the j^{th} letter within the template. To illustrate, suppose *wildcat* is presented across letter channels 3 to 11. In this case, the first letter projects to Channel 3 (i.e., IC = 3), and the resonating phase of the *cat* word is 4 (i.e., $\text{resPhase}_i = 4$). Thus, $c_j = 3 + 4 + \text{pos}_j - 1 = \text{pos}_j + 6$. That is, the channel that codes the first letter of the template is $c_j = 1 + 6 = 7$. Hence the *cat* node sends a positive feedback signal to the *C* letter node in Channel 7. The strength of this feedback is

$$\text{wordLetterInput}_{cji}(t) = \frac{\alpha_{\text{WL}}[\text{act}_i(t)]^+}{\sum_j \text{featureLetterInput}_{cj}(t) + .001}, \tag{36a}$$

where $\text{featureLetterInput}_{cj}(t)$ is as defined in Equation 22. The division by featureLetterInput implies that top-down feedback has a relatively weaker impact on channels that are receiving very strong bottom-up input (i.e., top-down feedback tends not to override unambiguous bottom-up input, unless (α_{WL} is large), but has a large impact on channels that receive little (or no) bottom-up input. For letter nodes that do not receive feedback,

$$\text{wordLetterInput}_{cji}(t) = 0. \tag{36b}$$

Equations 17–36 complete the description of how the input coding and matching algorithm is embedded within a model of lexical activation and selection. These equations require 15 parameters in order to weight the various influences on letter and word nodes.

Simulating Lexical Decision

Each of the experiments simulated here used the lexical decision task, and thus it is important to describe how the model can make lexical decisions based on its internal states. For this purpose, the opponent process model of lexical decision (Davis, 1999) was used. According to this model, lexical decision involves a competition between two opposing channels, one that accumulates evidence in favor of a yes response and another that accumulates evidence in favor of a no response. A decision is reached once one of the two channels exceeds a threshold activity (this decision threshold was set at .8). The activity equations for these two channels are similar in form to the shunting activity equations assumed for letter and word nodes, thus,

$$\frac{d}{dt}\text{YES} = (1 - \text{YES})\,\text{yes}_{in} - (1 + \text{YES})\lambda[\text{NO}]^+; \tag{37}$$

$$\frac{d}{dt}\text{NO} = (1 - \text{NO})\,\text{no}_{in} - (1 + \text{NO})\lambda[\text{YES}]^+. \tag{38}$$

In Equations 37 and 38, the first term following the equals sign shunts the positive input to the channel and establishes an upper bound (of 1) on channel activity, whereas the final term shunts the negative input to the channel (i.e., the inhibitory signal from the competing channel) and establishes a lower bound (of −1) on channel activity. The parameter λ represents the strength of between-channel inhibition and was set at .003.

The present implementation includes two sources of input to the yes channel:

$$\text{yes}_{in}(t) = y_{\text{global}}\text{wordSum}(t) + y_{\text{id}}\text{ID}. \tag{39}$$

The first source of input to the yes channel is the total activity at the word level, weighted by the parameter y_{global}, which was set at .4. The wordSum term is simply the sum of the positive word node activities; this source of input represents a measure of the word-likeness of the stimulus. A second (and more reliable) source of input to the yes channel is the evidence that lexical identification has occurred. When a word node exceeds some identification threshold μ (a threshold of .68 is used in the present simulations), it outputs a signal to the next level in the processing hierarchy; the summed activity at this level therefore provides an index of word identification having occurred. This higher processing level is not shown in Figure 6, and it is not explicitly modeled in the present simulations. Instead, the ID term in Equation 39 is used as a proxy for the total activity at this level; thus, ID = 1 if a word node has exceeded the identification threshold, and ID = 0 otherwise. The

parameter y_{id} weights the contribution of lexical identification to the yes decision. A large value implies that the yes channel will exceed the decision threshold (triggering a yes decision) soon after lexical identification; all simulations reported here used a value of $y_{id} = 1$. With the present parameter settings, it typically takes around 20 cycles following word identification before a yes response is triggered.

The present implementation includes only one source of input to the no channel:

$$no_{in}(t) = n_{letter}\max[act_{cj}(t)]. \quad (40)$$

This input represents the maximum activity at the letter level, weighted by the parameter n_{letter}, which was set at .36. Equation 40 implies that the decision process starts as soon there is activity in the letter nodes (and not, for example, when the stimulus is a forward mask such as ######). Furthermore, Equations 37–40 imply that the rate at which activity in the two channels grows varies according to the rate of letter activation. If, for example, stimulus degradation causes letter activation (and hence, also, word activation) to grow relatively slowly, this rate modulation ensures that the model will not say no prematurely.

The opponent process model has various advantages over the variable deadline models that have been used in previous modeling of the lexical decision task (e.g., Coltheart et al., 2001; Grainger & Jacobs, 1996), notably, with respect to explaining the continuous (rather than discrete) nature of reaction time distributions for no responses (Davis, 1999). The model has yet to be applied to detailed modeling of reaction time distributions (e.g., Norris, 2009; Ratcliff, Gomez, & McKoon, 2004), although Davis (1999) conducted simulations showing that the addition of noise to the yes_{in} and no_{in} equations results in positively skewed distributions like those observed in human data and observed that this skew is greater for low frequency words than for high frequency words (e.g., Andrews & Heathcote, 2001; Balota & Spieler, 1999).

Parameter Settings

Most of the parameters of the model are inherited from the original IA model. The settings for these parameters are very similar to (in most cases, identical to) those used in previous IA simulations (Davis, 2003; Davis & Lupker, 2006; McClelland & Rumelhart, 1981), with the exception that the feature-letter parameters have higher values, so that the speed of letter level activation, relative to the speed of word level activation, is considerably faster than in the original model. This parameter choice, combined with the step size parameter *dt* (i.e., the width of the time slices used to update activities in the model, which was set to .05), was chosen so that the scale of priming effects in the model (measured in processing cycles) was comparable with the scale of empirical priming effects (measured in milliseconds). The full list of parameter settings is shown in Table 3. For convenience, the parameters are listed in the order described in the text, together with references to the equations in which they are introduced. Note that σ parameters weight excitatory inputs, whereas γ parameters weight inhibitory inputs.

The spatial coding model also introduces several new parameters that are not in the original IA model; the implicit values of these parameters in the original model are shown in parentheses in Table 2. The introduction of letter position uncertainty that increases linearly with the number of letters in the input stimulus requires two new parameters, σ_0 and κ_σ, as described in Equation 3. A further parameter (Power) is used to contrast-enhance the bottom-up input to word nodes (see Equation 25). In previous modeling, researchers have used values of 2 (Davis, 1999; Davis & Bowers, 2004, 2006) or 3 (Lupker & Davis, 2009) for this parameter; the present simulations adopted an intermediate value of 2.5. The modifications introduced to handle words of varying length require two new parameters: the masking field parameter w_{mf} (see Equations 28–31) and the length mismatch parameter γ_{len} (see Equation 34). The FreqBias parameter, which modulates the competitive advantage of higher frequency words (see Equation 18), was set at 1.8 (larger settings of this parameter result in larger frequency effects, but can lead to difficulty in identifying the lowest frequency words). Finally, the parameter DecayCutoff,

Table 3
Parameter Settings Used in the Spatial Coding Model (SCM) and in the Original Interactive Activation Model (IAM)

Parameter	SCM	IAM	Equation
σ_0	.48	(0)	(3)—Position uncertainty by length function
κ_σ	.24	(0)	(3)—Position uncertainty by length function
FreqScale	.046	.046	(17)—Scaling of word frequency in resting activities
FreqBias	1.8	(−0.07)	(18)—Resting activity input to activity equation
ActMin	−.2	−.2	(19)—Shunting of net input by current activity
DecayCutoff	.4	(1)	(20)—Match-dependent decay
DecayRate	1	.07	(20)—Match-dependent decay
α_{FL}	.28	.005	(23)—Feature-letter input
γ_{FL}	6	.15	(23)—Feature-letter input
α_{LW}	.4	.07	(25)—Net word input
Power	2.5	(1)	(25)—Net word input
γ_{LW}	.04	.04	(26)—Mismatch inhibition
γ_{WW}	.34	.21	(27)—Word–word inhibition
α_{WW}	.44	.21	(27)—Word–word excitation
w_{mf}	.35	(0)	(28)—Masking field weight
γ_{len}	.06	(0)	(34)—Length mismatch
α_{WL}	.3	.3	(36)—Word–letter feedback
dt	.05	.05	(18)—Step size: Temporal scaling parameter

Note. Parentheses around values indicate parameters that are not (explicitly) included in IAM. Freq = frequency; ActMin = activity minimum; FL = feature-letter; LW = letter-word; WW = word–word; mf = masking field; len = length; WL = word–letter.

which controls match-dependent decay (see Equation 20), was set at .4.

The opponent process model of lexical decision requires five parameters: an identification threshold μ, three parameters that weight the inputs to the yes and no channels (y_{global}, y_{id}, and n letter), and a parameter, λ, that controls the strength of the inhibition between channels. Given the relatively large weight assigned to the y_{id} parameter in the present simulations, the speed of yes responses for words was dictated largely by the speed of lexical identification (i.e., how long it took before the activity of a word node exceeded the identification threshold).

Part 2: Application of the Model to Empirical Phenomena

General Method for Running Masked Priming Simulations

The general method for conducting the simulations was identical to that used in previously reported simulations (Davis, 2003; Davis & Lupker, 2006; Lupker & Davis, 2009). (The software and stimulus files for running the simulations can be downloaded from this webpage: http://www.pc.rhul.ac.uk/staff/c.davis/SpatialCodingModel/). Each simulation used exactly the same stimuli as used in published experiments, except where English language stimuli replaced those used in the original French, Dutch, or Spanish language experiments, as explicitly noted. The parameter settings of the model were identical for the simulations of primed and unprimed lexical decision. It would have been possible to achieve a better fit to the data if parameters were allowed to vary across simulations, and in some cases, this parameter variation might be justifiable, given that the experiments simulated were run in different laboratories with different populations of subjects. However, the main goal of the simulations was not to maximize the fit between model and data but to test whether a single model could capture all of the key qualitative results in the empirical database of orthographic form priming effects while also capturing the benchmark results in unprimed lexical decision.

At the beginning of each trial, activities of all nodes in the model were set to their resting levels. The input stimulus was then presented to the model by setting the binary letter feature nodes to the appropriate values. In the masked priming simulations, this stimulus was the prime, and it was replaced at the feature level by the target stimulus after 55 cycles (the value of 55 is approximately equal to the mean, in milliseconds, of the prime durations used in the experiments simulated here). The use of a fixed prime duration facilitates comparison across simulations; any small variations in prime duration in the actual experiments are treated as noise (along with differences in stimulus luminance, participant populations, and testing equipment). The letter-reset assumption of Davis and Lupker (2006) was adopted, according to which the target onset has the effect of resetting letter-level activities, as well as the yes and no channels.

On each cycle of processing, the difference equation in Equations 18 was solved, causing activities of the letter and word nodes to be updated; likewise, numerical integration of Equations 37 and 38 caused the activities of the yes and no channels to be updated on each cycle. A decision was said to have been made once the activity in one of the latter channels exceeded the criterion of .8. Decision latencies were measured from target onset.

Vocabulary

The vocabulary of the model consisted of 30,605 words taken from the N-Watch program (Davis, 2005). This set comprises all of the words listed in the CELEX database (Baayen et al., 1993) that (a) are between two and 10 letters in length, (b) occur six or more times in the corpus, that is, have an expected occurrence of at least 0.34 per million words, and (c) do not include capitals in the database listing (e.g., proper nouns like *England* or *Chris* were excluded).

Simulating Unprimed Lexical Decision With the Model

The majority of the simulations presented here focus on modeling correct yes responses in the masked priming variant of the lexical decision task. However, before introducing these simulations, it is appropriate to present some results demonstrating that the model can explain some benchmark findings in unprimed lexical decision. I consider seven lexical decision findings that Coltheart et al. (2001, p. 227) identified as "benchmark results that any computational model of reading should be able to simulate."

The word frequency effect. The first benchmark finding noted by Coltheart et al. (2001), and probably the most well established finding in the lexical decision task, is the word frequency effect; that is, the finding that yes responses to high frequency words are faster than yes responses to low frequency words (e.g., Monsell, 1991; Murray & Forster, 2004; Rubenstein, Garfield, & Millikan, 1970). Coltheart et al. (2001) demonstrated the DRC model's ability to simulate a word frequency effect by reporting a simulation of the stimuli from Andrews (1989, 1992); here, I take the same approach in testing the spatial coding model. These stimuli consist of 24 low frequency words (with an average frequency of occurrence of 2.4 per million words according to the CELEX database) and 24 high frequency words (average frequency of 444.4 per million). The model responded yes to each of the words except for *mope*, which is not included in its vocabulary. The mean predicted latencies of correct yes responses were 95.6 cycles for high frequency words, and 113.4 cycles for low frequency words; this difference was statistically significant ($p < 10^{-16}$). The model also showed good predictive power at the item level: The correlation between the model's decision latency and the mean human decision latency for each item (reported in Appendix A of Andrews, 1992) was .71.

The lexical status effect. The second benchmark finding noted by Coltheart et al. (2001) is that correct yes responses are faster than correct no responses (e.g., Rubenstein et al., 1970). This result is readily simulated by the model provided that the no_{in} parameter is not set too high. By way of demonstration, I constructed a set of 48 nonwords by changing a single letter of each of the words from the Andrews (1989, 1992) set. The mean decision latency for these nonwords was 125 cycles (the range of correct no latencies was 102 cycles to 160 cycles; the model made two errors, misclassifying *knaw* and *fect* as words). This mean latency is significantly slower ($p < 10^{-6}$) than the mean of 104 cycles for the matched words, although there is some overlap in the

distributions (the range for correct yes responses was 89 cycles to 126 cycles).

The N effect on yes latencies. The third and fourth findings described by Coltheart et al. (2001) as benchmark results related to neighborhood size (N) and are (a) the facilitatory effect of N on yes latencies to low frequency words and (b) the null effect of N on yes latencies to high frequency words. As has been noted previously (e.g., Bowers et al., 2005b; Davis, 2003), competitive network models such as the IA model and the spatial coding model predict a null effect of N on the speed of word identification (or more precisely, the models predict no difference between small-N words and large-N words, other things being equal; they do predict an inhibitory effect of having one or two neighbors relative to words with no neighbors). For example, the simulation of the Andrews (1989, 1992) stimuli predicts no difference between the large-N condition and the small-N condition ($p = .33$).

One approach to making competitive network models predict a facilitatory effect of N is to assume that fast yes responses are sometimes made prior to word identification (e.g., Coltheart et al., 2001; Grainger & Jacobs, 1996). This approach makes sense in experiments in which N systematically distinguishes words from nonwords. However, an alternative approach is to question the status of the facilitatory N effect. As Stadthagen-Gonzalez and Davis (2006) noted, N is positively correlated with imageability and negatively correlated with age-of-acquisition (AoA; i.e., large-N words tend to be learned earlier than small-N words). Although both of these variables are known to have large effects on lexical decision latency (e.g., Balota, Cortese, Sergent-Marshall, Spieler, & Yap, 2004; Brysbaert & Ghyselinck, 2006; Cortese & Khanna, 2007; Stadthagen-Gonzalez, Bowers, & Damian, 2004; Whaley, 1978) and reading time (Juhasz & Rayner, 2003), published experiments on N effects have not been controlled for their effects, and manipulations of N have typically been confounded with AoA and/or imageability (for example, the low frequency small-N and large-N words in the Andrews stimuli are typical in differing significantly with respect to both AoA and imageability ($p < .0005$ and $p < .05$, respectively), based on the norms collected by Cortese & Khanna, 2008, and Cortese & Fugett, 2004). In unpublished experiments, Davis and Bowers (2010) found no effect of N on the latency of yes responses when AoA and imageability were both controlled. Thus, I claim that the model's prediction of no effect of N on yes responses is the correct one.

The N effect on no latencies. The fifth benchmark finding noted by Coltheart et al. (2001) is the inhibitory effect of N on no latencies to nonwords, a finding first reported by Coltheart et al. (1977). Forster and Shen (1996) parametrically manipulated N and found that no response latency increased linearly with nonword N. It may be noted in passing that the latter result cannot be simulated by the model described by Coltheart et al. (2001) because the decision deadline procedure assumed in that model produces only two possible latencies for no responses (one for easy nonwords and another for more wordlike nonwords). I conducted a simulation of the spatial coding model using Forster and Shen's (1996) stimuli, which consisted of 120 items split into four N conditions (roughly N = 0, 1, 2, and 4). The model responded yes to three items: *millet, garter* (which are words in the model's vocabulary), and *forver*. As can be seen in Figure 7, the mean correct latencies showed a similar linear effect to that observed in the human data.

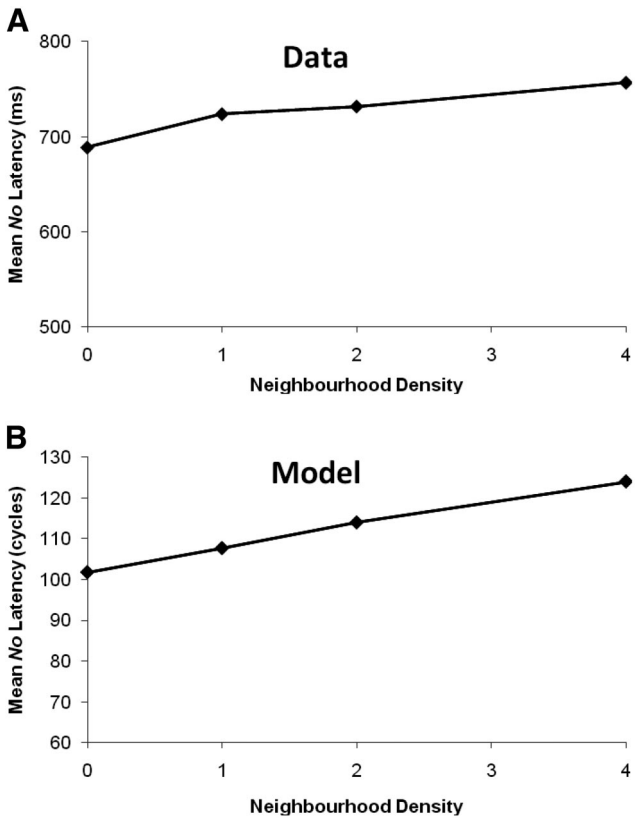

Figure 7. A: Data from Forster and Shen (1996) showing the effect of nonword N on the latency of no responses. B: Simulation results for the model tested on the same stimuli.

The pseudohomophone effect. The final two benchmark findings noted by Coltheart et al. (2001) are the pseudohomophone effect (Rubenstein, Lewis, & Rubenstein, 1971) and the interaction of this effect with orthographic similarity to the base word (Coltheart & Coltheart, as cited in Coltheart et al., 2001). As Coltheart et al. (2001, p. 231) noted, the DRC is able to simulate these effects by virtue of "feedback to the orthographic lexicon through the following route: letters to GPC rules to phoneme level to phonological lexicon to orthographic lexicon." The spatial coding model does not incorporate such a route and cannot explain the pseudohomophone effect. In principle, however, the addition of such feedback would enable the model to capture these two findings. For a demonstration of how these effects can be simulated by an orthographic model by assuming external phonological feedback, see Davis (1999; Simulation 6.3).

The nonword legality effect. Another basic finding concerning lexical decisions to nonwords is that illegal nonwords like *glazb* can be rejected more rapidly than can legal nonwords like *drilk* (Rubenstein et al., 1971). Presumably there is a phonological contribution to this effect, and thus, the present model should not be expected to provide a complete account of the illegality effect. Nevertheless, a simulation with Rubenstein et al.'s (1971) stimuli showed that the model's latencies exhibit a significant effect in the same direction, with mean no latencies of 102.3 cycles for illegal nonwords and 124.1 cycles for legal nonwords ($p = .001$).

In summary, the spatial coding model captures a number of basic lexical decision findings, including the word frequency effect, the lexical status effect, the nonword illegality effect, and the inhibitory effect of nonword N, and extensions of the model to include phonological processing would enable it to capture the pseudohomophone effect. Other simulations of the model (not reported here) have shown that it predicts the inhibitory effect of higher frequency neighbors (e.g., Grainger, O'Regan, Jacobs, & Segui, 1989), including the effects of addition and deletion neighbors (e.g., Davis et al., 2009; Davis & Taft, 2005).

Ability of the Model to Predict Item-Level Variance

Another way to evaluate computational models of visual word recognition is to examine how well these models predict performance at the level of individual items. The development in recent years of megadatabases of lexical decision and naming latencies (e.g., Balota et al., 2004; Spieler & Balota, 1997) has facilitated such evaluations. Initial findings based on this approach were not especially promising. Spieler and Balota (1997) collected naming latencies for approximately 2,870 monosyllabic words and found that the orthographic error scores from the Seidenberg and McClelland (1989) model predicted 10.1% of the variance, whereas the settling times from the Plaut et al. (1996) model predicted just 3.3% of the variance in the human naming latencies. Subsequently, Coltheart et al. (2001) reported that the DRC model accounted for 3.5% of the variance for a subset of 2,516 words from this database. Clearly, predicting performance at the level of individual items is a rather stringent test of computational models.

To compare the predictive power of the spatial coding model, I used exactly the same set of monosyllabic words as a test set, excluding a small proportion of items that were not included in the model's vocabulary (very low frequency words like *awn*); the resulting set consisted of 2,715 words. Mean lexical decision latencies for these items were obtained from the English Lexicon Project (Balota et al., 2004). The correlation between the model's lexical decision latencies and the human data was .51; that is, the model successfully accounts for 26% of the variance at the level of individual items. It is also interesting to examine the naming latencies for this set. Strictly speaking, the spatial coding model is not able to name words because it has no phonological output units. Nevertheless, it could be used to provide the lexical route of a dual-route model of word naming, and thus, to the extent that there is a lexical contribution to word naming latency, the model should have some predictive power. The model's latencies predicted 10.2% of the variance in the naming latencies from the English Lexicon Project. In summary, although much of the variance in human lexical decision and naming latencies remains unaccounted for, the spatial coding model appears to be doing at least as well as other notable models of visual word recognition.

Masked Form Priming Simulations

Organization of simulations. The masked priming technique has been used to study many different aspects of visual word recognition, including orthographic, phonological, morphological, and semantic processes. Given that the model under consideration does not incorporate phonological, morphological, or semantic representations or processes, I did not seek to simulate experiments that specifically focus on these processes. Despite this restriction, the relevant database of masked priming results is quite large. Fortunately, the main results in this domain are fairly well established. As Grainger (2008, p. 9) noted, "Perhaps the most stable, replicable, and therefore uncontroversial results obtained with the masked priming paradigm concern purely orthographic manipulations."

The simulations below are arranged thematically into subsections according to the type of letter string manipulation: replacement, transposition, deletion, insertion, string reversal, or string displacement. Multiple experiments are simulated within each subsection. The relevant data from these experiments, together with the predicted priming effects, are summarized in Table 4. Attempting to capture the pattern of priming effects associated with each of these string manipulations imposes very strict constraints on any model of orthographic input coding. As will be seen, the spatial coding model handles this challenge quite successfully.

A. Primes that involve letter replacement.

Simulation 1: The prime lexicality effect (Davis & Lupker, 2006, Experiment 1). There are many experiments demonstrating facilitatory neighbor priming effects, for example, that responses to the target *SHIRT* are faster following the one-letter different prime *shint* relative to an unrelated prime like *cland* (e.g., Davis & Lupker, 2006; Forster et al., 1987; van Heuven, Dijkstra, Grainger, & Schriefers, 2001). However, not all one-letter different primes result in facilitatory priming. When the prime is itself a word (e.g., *short–SHIRT*), responses to the target are slower, relative to control word primes (e.g., Segui & Grainger, 1990). This pattern of facilitatory priming from nonword neighbors and inhibitory priming from word neighbors was demonstrated for the same set of targets in a lexical decision experiment reported by Davis and Lupker (2006). That is, classifications of a target like *AXLE* were facilitated by the nonword neighbor prime *ixle* (relative to an unrelated nonword prime) but inhibited by the word neighbor prime *able* (relative to an unrelated word prime like *door*). Inhibitory priming effects were largest when the prime was of high frequency and the target was of low frequency (e.g., inhibition tended to be stronger for *able–AXLE* than for *axle–ABLE*). This effect can be observed in the mean latencies shown in Table 4.

Results like those reported by Davis and Lupker (2006) are important for understanding masked form priming effects, as they reveal how such priming effects are subject to both excitatory and inhibitory influences. Indeed, being able to simulate the results of this experiment can be viewed as a prerequisite for each of the simulations that follow, in which the excitatory and inhibitory influences of masked primes are systematically varied. Simulation 1 sought to test the spatial coding model's ability to predict the prime lexicality effect. The simulation tested the same set of four and five-letter words used by Davis and Lupker (2006). The basic procedure followed that described in the General Method for Running Masked Priming Simulations section, above.

The results of the simulation are shown in Table 4. As can be seen, the model does a good job of capturing the prime lexicality effect that was observed by Davis and Lupker (2006). That is, related word primes produce inhibitory priming effects, whereas related nonword primes produce facilitatory priming effects. The model also captures the greater inhibitory priming for low fre-

Table 4
Results of Twenty Masked Priming Simulations, Together With the Original Experiment Means

Index	Simulation, experiment, and prime type	Data			Model			Difference
		Critical	Control	Effect	Critical	Control	Effect	
	Simulation 1							
	Davis & Lupker (2006) Experiment 1							
1	HF primes, LF targets	679	645	*−34*	150	115	*−35*	−1
2	Nonword primes, LF targets	634	660	*26*	93	113	*20*	−6
3	LF primes, HF targets	586	573	*−13*	124	103	*−21*	−8
4	Nonword primes, HF targets	571	582	*11*	79	101	*23*	12
	Simulation 2							
	van Heuven et al. (2001)							
5	Shared neighbor prime	540	552	*12*	97	104	*7*	−5
6	No shared neighbor prime	524	552	*28*	84	104	*20*	−8
	Simulation 3							
	Schoonbaert & Grainger (2004) Experiment 4							
7	Replace initial letters	622	623	*1*	102	104	*2*	1
8	Replace initial letters	626	623	*−3*	105	104	*0*	3
9	Replace final letters	620	623	*3*	103	104	*1*	−2
	Simulation 4							
	Lupker & Davis (2009) Experiment 2A (standard priming)							
10	Replace 1	496	518	*22*	70	102	*32*	10
11	Replace 2	501	518	*17*	89	102	*13*	−4
12	Replace 3	517	518	*1*	99	102	*3*	2
13	Replace 4	514	518	*4*	99	102	*2*	−2
14	Replace 5	525	518	*−7*	100	102	*1*	8
	Simulation 5							
	Lupker & Davis (2009) Experiment 2B (sandwich priming)							
15	Replace 1	552	609	*57*	41	99	*58*	1
16	Replace 2	576	609	*33*	56	99	*43*	10
17	Replace 3	582	609	*27*	72	99	*28*	1
18	Replace 4	601	609	*8*	84	99	*15*	7
19	Replace 5	602	609	*7*	94	99	*5*	−2
	Simulation 6							
	Perea & Lupker (2003b) Experiment 1							
20	Identity	523	570	*47*	56	108	*53*	6
21	Internal transposition	556	586	*30*	76	106	*31*	1
22	Final transposition	554	567	*13*	92	104	*12*	−1
	Simulation 7							
	Davis & Bowers (2006) Experiment 2 & 3							
23	SN2	584	613	*30*	79	107	*28*	−2
24	SN4	582	613	*31*	78	107	*29*	−2
25	N1R−	595	613	*19*	86	107	*21*	2
26	N1R+	595	613	*18*	86	107	*21*	3
	Simulation 8							
	Perea & Lupker (2004) Experiment 1B							
27	SN	665	703	*38*	76	106	*30*	−8
28	NATN	679	703	*24*	81	106	*24*	0
29	DSN	696	703	*7*	97	106	*9*	2
	Simulation 9							
	Guerrera & Forster (2008) Experiment 3A							
30	T-I-6	610	636	*26*	78	109	*30*	4
31	T-all	637	636	*−1*	104	109	5	6
	Lupker & Davis (2009) Experiment 1A (standard priming)							
32	T-all	718	727	*9*	104	109	*5*	−4
	Simulation 10							
	Lupker & Davis (2009) Experiment 1B (sandwich priming)							
33	T-all	651	691	*40*	76	106	*30*	−10
	Simulation 11							
	Guerrera & Forster (2008) Experiment 3A							
34	Reversed halves	637	636	*−1*	105	109	*3*	4
	Simulation 12							
	Davis & Lupker (2009) Experiment 2A (standard priming)							
35	Reversed Interior	620	625	*5*	102	105	*3*	−2

Table 4 (*continued*)

Index	Simulation, experiment, and prime type	Data			Model			Difference
		Critical	Control	Effect	Critical	Control	Effect	
	Simulation 13							
	Davis & Lupker (2009) Experiment 1B (sandwich priming)							
36	Reversed halves	684	693	*9*	100	106	*6*	−3
	Davis & Lupker (2009) Experiment 2B (sandwich priming)							
37	Reversed interior	643	666	*23*	80	102	*21*	−2
	Simulation 14							
	Welvaert et al. (2008)							
38	Insert 0	545	600	*55*	57	107	*50*	−5
39	Insert 1	556	600	*44*	69	107	*38*	−6
40	Insert 2	567	600	*33*	81	107	*26*	−7
41	Insert 3	578	600	*22*	89	107	*18*	−4
	Simulation 15							
	Van Assche & Grainger (2006) Experiment 1							
42	1234567	532	582	*50*	57	107	*50*	0
43	12334567, 12345567	541	582	*41*	65	107	*42*	1
44	12534567, 12345367	543	582	*39*	66	107	*42*	3
45	12d34567, 12345d67	543	582	*39*	65	107	*42*	3
	Simulation 16							
	Schoonbaert & Grainger (2004) Experiment 1							
46	Repeated letter deletion	565	607	*42*	74	106	*33*	−9
47	Unique letter deletion	572	607	*35*	75	106	*31*	−4
48	Repeated letter control	560	597	*37*	74	107	*33*	−4
49	Unique letter control	561	597	*36*	76	107	*32*	−4
	Simulation 17							
	Peressotti & Grainger (1999) Experiment 2							
50	1346	623	650	*27*	82	103	*21*	−6
51	1d34d6	629	641	*12*	90	104	*13*	1
	Simulation 18							
	Peressotti & Grainger (1999) Experiment 3							
52	1346	596	616	*20*	82	103	*21*	1
53	1436	611	616	*5*	91	103	*12*	7
54	6341	609	616	*7*	101	103	*1*	−6
	Simulation 19							
	Kinoshita & Norris (2009)							
55	Shifted halves	603	610	*7*	108	108	*1*	−6
	Simulation 20							
	Grainger et al. (2006)							
56	12345	531	576	*45*	74	101	*28*	−17
57	34567	539	576	*37*	74	101	*27*	−10
58	13457	547	576	*29*	*78*	101	*23*	−6
59	1234	562	585	*23*	88	101	*13*	−10
60	4567	573	585	*12*	93	101	*8*	*−4*
61	1357	577	585	*8*	95	101	*6*	−2

Note. Italicized values are differences of the values in the preceding two columns. HF = high frequency, LF = low frequency, SN = substitution neighbor, N1R = neighbor once-removed, NATN = nonadjacent transposition neighbor, DSN = double substitution neighbor, T-I-6 = condition in which the exterior letters were maintained in their correct position but the six interior letters were each transposed with an adjacent letter, T-all = condition in which all eight letters of the target were transposed.

quency targets primed by high frequency words than for high frequency targets primed by low frequency words. Finally, the model does not show the frequency interaction for facilitatory priming that was observed by Davis and Lupker (2006). However, this interaction was not statistically significant, and the 11 ms priming effect observed in the human data for high frequency targets may be an underestimate of the true effect.

The outcome of this simulation effectively replicates Davis and Lupker's (2006) finding that a (modified) IA model with slot coding could simulate their results. However, there are important differences between the model tested by Davis and Lupker (2006) and the model tested here. In the former simulation, the four-letter word stimuli were tested with a vocabulary of 1,178 words and the five-letter word stimuli were tested with a separate vocabulary of 3,370 words. By contrast, the present simulation tested all of the stimuli with a fixed vocabulary over 25 times larger than that of the original IA model. Furthermore, switching to spatial coding introduces additional orthographic neighbors (e.g., transposition neighbors, neighbors once removed, etc.). The simulation results show that these changes in the structure of the lexical neighborhood do not affect the model's ability to simulate the basic prime lexicality effect.

There is a far more important implication of these data. It would be overly simplistic to draw the conclusion that related word primes have inhibitory influences, whereas related nonword primes have facilitatory influences. Rather, what these empirical

data and simulation results show is that masked form priming effects reflect the combination of facilitatory and inhibitory influences. In the case of word primes (especially high frequency word primes), the inhibitory influences typically overwhelm the facilitatory influences, whereas in the case of nonword primes the opposite is true. Nevertheless, it is critical to note that inhibitory influences on masked priming are always present to the extent that the prime activates competitors of the target, whatever the lexical status of the prime. This insight has major implications for all of the form priming experiments simulated in this article. In particular, it implies that the match values computed by models of orthographic input coding are only part of the story. These match values drive the facilitatory influence of the prime on the target. But to make accurate quantitative predictions, it is necessary to also take into consideration the inhibitory influences of the prime, which requires conducting simulations of a full model of lexical identification. Predictions based purely on match values fail to capture these inhibitory influences of lexical competitors. For example, the nonword *blard* and the word *board* result in equivalent (large) match values with the target *BEARD*. This equivalence may suggest that the unmediated facilitatory influence of these primes on the target will be the same, but (as is clear from the results of the experiment and the simulation) their inhibitory influences on the target differ greatly. Understanding this point allows one to realize why masked priming experiments sometimes fail to observe facilitation for primes that are associated with relatively high match values (e.g., Guerrera & Forster, 2008; Schoonbaert & Grainger, 2004). It also enables one to devise methods for overcoming these inhibitory influences, as discussed below.

Simulation 2: The shared neighborhood effect (van Heuven, Dijkstra, Grainger, & Schriefers, 2001). Another empirical phenomenon that illustrates the influence of lexical competitors on masked form priming is the shared neighborhood effect reported by van Heuven et al. (2001). Shared neighbors are words that are neighbors of both the prime and the target. For example, in the case of the prime–target pair *laby–LAZY*, the word *lady* is a shared neighbor. Van Heuven et al. (2001) found that form priming effects were smaller when the prime and target shared a neighbor (as in *laby–LAZY*), compared with trials when there were no shared neighbors (e.g., *lozy–LAZY*).

Simulation 2 attempted to simulate this finding. The stimuli were constructed to parallel the Dutch stimuli used by van Heuven et al. (2001). There were 20 four-letter, low frequency, small-N target words. For each target, three primes were constructed: one that shared one or more neighbors with the target, another that was a neighbor of the target but shared no neighbors with it, and a third that was an unrelated nonword prime. Position of the replacement letter was roughly matched across the two neighbor conditions.

The simulation slightly underestimated the magnitude of the observed facilitatory priming effects but captured the critical shared neighborhood effect in van Heuven et al.'s (2001) experiment. That is, the priming effect was substantially greater for primes that did not share any neighbors with the target than for primes that shared a neighbor with the target. The results of this simulation reinforce and strengthen the conclusion drawn from Simulation 1. In this case, the critical primes are all nonwords, so the results cannot be attributed to a simple prime lexicality account. The two sets of related primes were equally similar to their targets, and yet, one set produces a much larger priming effect than does the other. Here, the facilitatory influence of bottom-up input is matched, but the inhibitory influences of lateral inhibition are not: Shared neighbor primes give rise to activity in the target's lexical competitors, and the resulting competition diminishes the facilitatory influences of the prime. Consequently, as in Simulation 1, the results of the simulation cannot be determined purely on the basis of the orthographic match between the primes and the targets.

Related shared neighborhood results have been reported with word primes by Davis and Lupker (2006, Experiment 3) and with partial word primes (e.g., *c#be–CUBE*) by Hinton, Liversedge, and Underwood (1998) and J. R. Perry, Lupker, and Davis (2008). Each of these findings can be interpreted by noting that all primes have both facilitatory and inhibitory influences. This insight is also relevant for the following simulations.

Simulation 3: The multiple-letter replacement constraint (Schoonbaert & Grainger, 2004, Experiment 4). As noted already (and simulated in Simulation 1), form primes constructed by replacing a single letter of the target are typically associated with relatively large priming effects. However, when two letters are replaced, priming effects are greatly diminished and often absent (e.g., Perea & Lupker, 2003b, 2004; Peressotti & Grainger, 1999; Schoonbaert & Grainger, 2004). An experiment reported by Schoonbaert and Grainger (2004, Experiment 4) provides a good illustration of this apparent limit on form priming effects. Although there was some evidence of form priming from two-letter different primes in the case of seven-letter targets (as also observed by Lupker & Davis, 2009), there was no evidence at all of priming from two-letter different primes in the case of five-letter targets, regardless of whether the position of replacement was initial, medial, or final (see Table 4).

On the surface, this finding appears to pose a problem for all of the current orthographic input coding schemes, which predict that orthographic similarity values should decrease approximately linearly as more letters are substituted (at least for the replacement of successive internal letters). There is a reasonably high overlap between a five-letter target word and a two-letter different prime. For a pair like *BLEON* and *BARON*, the match value is $5/7 = .71$ (assuming dynamic end-letter marking). This match value is equivalent to the match computed for one-letter different primes where the different letter is an end letter, as in *BAROY* and *BARON*. Thus, if there was a straightforward relationship between predicted match values and observed priming effects, two-letter different primes like *prade–PROBE* should, according to the spatial coding model, produce priming effects that are at least as large as those associated with one-letter different primes like *baroy–BARON* (i.e., both primes should produce significant form priming).

From the foregoing discussion, however, it should be apparent that there is not a straightforward relationship between predicted match values and observed priming effects and that simulations are required to predict the priming effects that should be observed for two-letter different primes. To simulate Schoonbaert and Grainger's (2004) experiment, which was conducted with French stimuli, I selected the 14 five-letter target words from their stimulus set that are French–English cognates (such as *rural* and *baron*). Each target was paired with the same four primes used by Schoonbaert and Grainger (2004), that is, three separate two-letter different primes (initial, inner, and final letter replacements) and

an unrelated (all-letter-different) prime. For example, the primes for the target *BARON* were *upron* (initial two-letter different), *bleon* (inner two-letter different), *barsy* (final two-letter different), and *pievu* (control, all-letter-different).

The results of this simulation are shown in Table 4. As can be seen, the results for two-letter different primes mirrored the findings of Schoonbaert and Grainger (2004), in that they show no evidence of priming. In order to test whether the absence of priming could be due to characteristics of the targets, I constructed an additional one-letter different prime condition (i.e., one that was not included in the experiment) by replacing the final letter of the target, using the same replacement letter as for the two-letter different prime that incorporated a final letter replacement (e.g., the one-letter different prime for the target *BARON* was *baroy*). The mean decision latency for this condition was 93 cycles, that is, a priming effect of 11 cycles. Thus, these targets are capable of showing priming effects. Furthermore, although the match value computed for one-letter different primes with final letter replacements is equivalent to that computed for two-letter different primes with inner letter replacements (e.g., *bleon–BARON*), the latter condition showed no priming in the simulation. Thus, the absence of priming for two-letter different primes cannot be explained solely in terms of match values.

Why, then, is it that a prime like *bleon* does not (on average) facilitate responses to a two-letter different target like *BARON*? The reason for the absence of priming in the model in this case is the same as for the primes with shared neighbors in Simulation 2: The prime frequently activates competitors of the target more strongly than the target itself. In the case of two-letter different primes, these competitors are not quite as obvious, partly because psycholinguists since Coltheart et al. (1977) have tended to count only neighbors formed by a single letter replacement. Nevertheless, these competitors exist and exert an inhibitory influence on priming; for example, in the case of *bleon–BARON*, the lexical competitors will include words like *blown* (a neighbor once removed of the prime), *bacon*, *began*, *brown*, and *bean*. One way to establish that this is the correct explanation of the absence of priming in the model is to disable all of the word nodes for words that are shared neighbors (in the broad sense) of *bleon* and *BROWN* (e.g., *blown*, *bacon*, *began*, *baton*, *bison*, *begin*, *brown*, etc.). When this is done (defining a shared neighbor as any word that produces a match of greater than .4 with both the prime and the target), the priming effect for this trial changes from null to 10 cycles. Thus, according to the model, the multiple letter replacement constraint in masked priming is due to the fact that increases in the number of letters that differ between the prime and the target both decrease the orthographic match between these two stimuli (and increases the mismatch) and simultaneously increase the likelihood that the prime will activate lexical competitors of the target.

Relaxing the multiple-letter replacement constraint. As has been seen, prime–target pairs that have relatively high match values can have facilitatory, null, or inhibitory effects, depending on the extent to which they activate lexical competitors of the target. A technique for greatly reducing lexical competitor effects has recently been developed by Lupker and Davis (2009). In this technique, called sandwich priming, the prime of interest (e.g., a related form prime or an unrelated control prime) is preceded by a brief (masked) presentation of the target word; that is, the prime is sandwiched between two presentations of the target. The aim of the first presentation of the target stimulus is to give an initial headstart to the activation of the target node, enabling it to overwhelm lexical competitors that would ordinarily be activated by the prime. Thus, if the absence of priming for two-letter different primes in Simulation 3 was the result of lexical competitor effects, it should be possible to obtain form priming for these stimuli if the sandwich priming technique is used. Simulation 3A tested this prediction. Sandwich priming was simulated by presenting the target (for 40 cycles) prior to the prime of interest. In all other respects, this simulation was identical to Simulation 3. The results showed relatively large priming effects for two-letter different primes; the size of the priming effect was 25, 38, and 25 cycles for the initial, inner, and final letter replacement conditions, respectively. Thus, as expected, the use of sandwich priming causes the model to predict facilitatory effects for prime conditions that did not produce priming in Simulation 3. This in turn leads to an empirical prediction concerning the difference between conventional masked priming and sandwich priming. Although this prediction has not been tested for Schoonbaert and Grainger's (2004) stimuli, it has been tested by Lupker and Davis (2009) for seven-letter English target words; this test is the subject of the next pair of simulations.

Simulation 4: Parametric variation of number of replaced letters (Lupker and Davis, 2009, Experiment 2A). Although the replacement of two letters eliminates priming for five-letter word targets, the spatial coding model predicts that two-letter different primes should produce some priming for longer targets (the reason for this prediction is straightforward, for example, a match of 4/6 is greater than a match of 3/5). The available evidence supports this prediction, although there is some variability in the obtained effects. The average priming effect for two-letter different primes and five-letter targets, based on eight priming effects from four experiments (Frankish & Barnes, 2008; Perea & Lupker, 2003a, Experiments 1 and 2; Schoonbaert & Grainger, 2004, Experiment 4) is 0 ms (the median is 1 ms), whereas the average priming effect for two-letter different primes and six-letter targets, based on five priming effects from four experiments (Perea & Lupker, 2003b, Experiment 3; Perea & Lupker, 2004, Experiments 1 and 2; Peressotti & Grainger, 1999, Experiment 2) is 13 ms (the median is 12 ms). Although only two of the latter five priming effects were statistically significant, it seems likely that there is a genuine priming effect here.

Lupker and Davis (2009) examined priming for seven-letter targets, and parametrically manipulated the number of replaced letters between one and five. Results are shown in Table 4. As can be seen, the results showed clear priming effects (which were statistically significant) for one- and two-letter different primes but no priming for primes in which three or more letters of the target were replaced. Simulation 4 tested whether this result is captured by the model, with the same stimuli as in Lupker and Davis's (2009) experiment. As can be seen in Table 4, there was a relatively good match between the observed data and the results of the simulation, although the model overestimated the observed priming effect for one-letter different primes. The other slight discrepancy between the model and the data was for the five-letter different primes, but the −7 ms priming effect in the data is most likely attributed to noise. In summary, the model does a good job

of capturing the effects of letter replacement on masked form priming.

Simulation 5: Parametric variation of number of replaced letters (Lupker and Davis, 2009, Experiment 2B). In a separate experiment, Lupker and Davis (2009, Experiment 2B) tested the same stimuli with the sandwich priming technique. As Table 4 shows, the use of sandwich priming enabled significant priming to be obtained even when the prime and target differed by three (out of seven) letters. Simulation 5 tested whether this result is captured by the model; it was identical to Simulation 4 except for the initial presentation of the target for 40 cycles to simulate sandwich priming. As can be seen in Table 4, there was a good match between the observed data and the results of the simulation.

Summary of Simulations 1–5. The masked priming simulations presented above each deal with situations in which one or more letters of the target are replaced by other letters to form an orthographically similar prime. The match calculations in such cases are quite straightforward, but the simulations illustrate that masked priming effects are more complex than a simple account based on match values alone. Different prime–target pairs that are associated with identical match values can result in facilitatory, inhibitory, or null priming effects. Primes that are less similar to the target can produce greater facilitation than those that are more similar.

The cause of this additional complexity is lexical competition. In the most extreme case, as demonstrated in Simulation 1, a prime that is orthographically very similar to the target can nevertheless lead to inhibitory priming if the prime is itself a word, thereby promoting strong lexical competition. Corresponding, though less extreme, instances of lexical competition are observed for primes that are not themselves words but that activate words that can compete with the target. Such primes may fail to produce facilitatory priming of the target, even though other primes matching the target equally well produce facilitation, as in Simulations 2 and 3.

Simulations of lexical identification are essential for modeling these interactions of orthographic similarity and lexical competition; models that rely solely on match calculation cannot hope to provide an adequate account of form priming data. A further benefit of a computational approach that attempts to model identification processes during masked priming is that it can be used to suggest new priming methodologies. The sandwich priming technique illustrated in Simulation 5 is a case in point. This technique was initially tested computationally, where it was shown to have the potential to reduce lexical competitor effects. Subsequent empirical tests of the technique have validated this claim and established the usefulness of sandwich priming for obtaining priming effects in situations where lexical competition would ordinarily interfere with form priming (Lupker & Davis, 2009). Some of the simulations presented below will illustrate the application of this technique further. In particular, the value of techniques for reducing lexical competition will become apparent when extreme letter transpositions are considered.

B. Primes that involve letter transpositions. A number of early articles investigated TL similarity effects in unprimed tasks (e.g., Andrews, 1996; Bruner & O'Dowd, 1958; Chambers, 1979; O'Connor & Forster 1981; Taft & van Graan, 1998). However, in recent times, researchers have tended to favor the masked form priming procedure for investigating TL similarity effects. A search of PsycInfo reveals 25 published articles investigating TL priming in the last 5 years (several more are currently in press or under submission), and most of these articles report multiple experiments. Simulation 6 simulates an experiment from one of the early and particularly influential studies reported in this series of articles on TL priming (Perea & Lupker, 2003a). In Simulation 7, I examine what happens when a letter transposition is combined with a letter replacement, as in the experiments of Davis and Bowers (2006). The latter data enabled the falsification of a number of alternative models of orthographic input coding.

In the remaining simulations in this subsection, I investigate increasingly severe disruptions of letter position. In Simulation 8, I examine priming effects for primes in which the TLs are not immediately adjacent (as in Simulation 6) but are instead separated by an intervening letter. Experiments have shown that TL priming effects are obtained when the TLs are nonadjacent, as in *caniso–CASINO* (Davis & Bowers, 2005; Lupker, Perea, & Davis, 2008; Perea & Lupker, 2004), although these effects are smaller than the priming effects for adjacent TL primes. Finally, Simulations 9 and 10 investigate the effects of more extreme letter transpositions: Simulation 9 simulates the results of Guerrera and Forster (2008, Experiment 3), whereas Simulation 10 makes a prediction concerning the outcome of an experiment with the same primes with the sandwich priming methodology. The latter prediction has recently been confirmed by Lupker and Davis (2009).

Simulation 6: Adjacent TL priming (Perea & Lupker, 2003a). The goal in Simulation 6 was to investigate the ability of the spatial coding model to capture adjacent TL priming effects. Data from Experiment 1 of Perea and Lupker (2003a) provides an appropriate test set for this purpose. Their experiment used five-letter English words as target stimuli. In addition to manipulating prime type, Perea and Lupker (2003a) also manipulated position of transposition: internal (e.g., *jugde–JUDGE*) or final (e.g., *judge–JUDGE*). The stimuli in this experiment were carefully controlled, with orthographic controls for each of the TL primes. These control stimuli were constructed by making letter substitutions for the letters that were transposed in the TL primes; for example, different participants saw the target *GLOVE* preceded by a TL-internal prime (*golve*), a corresponding orthographic control (*gatve*), a TL-final prime (*gloev*), or its corresponding orthographic control (*gloac*).

Table 4 shows the results of the experiment and the simulation. As can be seen, there was a very close fit between model and data. There was a strong overall TL priming effect, as measured relative to orthographic controls. This TL priming effect was especially large for internal transpositions, although the priming effect was smaller than for identity primes (suggesting that letter position does matter, though this comparison of priming effects must be made with some caution because the identity priming effect was measured relative to all-letter-different primes rather than two-letter different orthographic controls). Priming was somewhat smaller for final transpositions. The results of the simulation reflect the relative flexibility of letter position coding when spatial coding is used, whereas the weaker TL priming for final transpositions is the result of end-letter marking.

Simulation 7: Neighbor once-removed priming (Davis & Bowers, 2006, Experiment 2–3). TL priming effects such as those observed in Simulation 6 (and empirically in many experiments) demonstrate the flexibility of the visual word identification system with respect to letter position coding. This flexibility raises the

question of just how sensitive this system is to letter position. A possibility left open by the basic TL priming effect is that the system is so flexible because it is actually somewhat insensitive to disruptions of letter position. One way to test this possibility is to investigate the system's sensitivity to the smallest possible disruption of letter position. This was the approach taken by Davis and Bowers (2006).

Davis and Bowers (2006) compared two types of form primes: neighbor primes, formed by substituting one of the letters of the target with a different letter, and neighbor once-removed primes, formed by combining a letter transposition with a substitution of one of the TLs. For example, for the target word *ANKLE*, *axkle* is a neighbor prime, whereas *akxle* is a neighbor once-removed prime. Note that both of these primes contain four letters in common with the target. The only difference, with respect to their similarity to the target, is that one letter (in this case, the *k*) is in the correct position in the neighbor prime but is one position away from the correct position in the neighbor once-removed prime. Given such a minimal difference, it would not seem implausible to posit that these two primes could be indistinguishable from the perspective of the target word node, and in fact, this is the prediction of the discrete open-bigram coding model (Grainger & van Heuven, 2003). However, Davis and Bowers (2006) found that this prediction is incorrect—neighbor primes were slightly, but significantly, more effective than neighbor once-removed primes. Davis and Bowers (2006) noted that the spatial coding model predicts higher match values for neighbor primes than for neighbor once-removed primes. However, they did not report simulations of masked priming. Simulation 7 was therefore designed to test the model's ability to simulate the empirically observed pattern.

The simulation tested the same set of 120 five-letter English target words used by Davis and Bowers (2006). Each target was associated with five primes. Two of these were neighbor primes in which the letter in either Position 2 or Position 4 of the target had been replaced (rows 23 and 24, respectively, in Table 4). There were also two neighbor once-removed prime conditions in which the letter in either Position 4 or Position 2 of the target was replaced and then transposed with the letter in Position 3 (rows 25 and 26, respectively, in Table 4). The final condition comprised unrelated control primes. As can be seen in Table 4, both types of form prime resulted in facilitatory priming, as in the experimental data, and the priming effect was larger for neighbor primes than for neighbor once-removed primes. The model provides a good fit to both the qualitative and quantitative pattern of the data. One further aspect of the results worth noting is that the model predicts no effect of the serial position of the replaced letter for neighbor primes (i.e., it did not matter whether the neighbor primes were formed by substituting the second or the fourth letter). This prediction agrees with the pattern observed by Davis and Bowers (2006). Although there may be differences between exterior letters and interior letters, there is apparently no difference in the status of different interior letters.

The ability of both readers and the model to show differences between neighbor and neighbor once-removed primes demonstrates the exquisite sensitivity of position coding in the visual word identification system. This sensitivity poses a challenge to many open-bigram coding models (Grainger & van Heuven, 2003; Whitney, 2001, 2004), although the overlap open-bigram model (Grainger et al., 2006) does predict greater match values for neighbor than for neighbor once-removed primes. At the same time, the fact that neighbor once-removed primes are associated with significant form priming, in contrast to the weak or null priming effects associated with two-letter different primes (Simulation 3), provides further evidence of the relative flexibility of letter position coding (i.e., the letter *k* in *akxle* contributes toward the match with *ankle*, despite being in the incorrect position). The next three simulations probe the limits of this flexibility.

Simulation 8: Nonadjacent TL Priming (Perea & Lupker, 2004). Perea and Lupker (2004) reported data showing that nonadjacent TL primes like *caniso–CASINO* produce greater form priming than orthographic control primes like *caviro–CASINO*. This finding, which has subsequently been replicated in English (Davis & Bowers, 2005; Lupker et al., 2008), demonstrates that the position uncertainty in the orthographic input code cannot be captured by a position-specific model in which letter inputs are assumed to "leak" into immediately adjacent channels. Perea and Lupker (2004) also found (in three separate lexical decision experiments) that the priming for nonadjacent TL primes was weaker than for neighbor primes (e.g., *casiro–CASINO*). This result, which has also been replicated in English (Davis & Bowers, 2005), provides even stronger constraints on models of letter position coding. Earlier open-bigram coding models predict larger match values for nonadjacent TL primes than for neighbor primes (e.g., Grainger & van Heuven, 2003; Whitney, 2001), although more recent open-bigram models can predict the opposite ordering, given appropriate parameter choices (Grainger et al., 2006; Whitney, 2004). The spatial coding model predicts the correct ordering of match values, although as has been seen already, this does not guarantee that the model will make the correct predictions regarding the outcome of the masked form priming experiment. Simulation 8 was designed to test the model's ability to simulate the empirical pattern.

The simulation tested a set of 102 six-letter English target words; these words had an average frequency of 32 per million and contained no repeated letters. The nonadjacent TL primes were created by transposing the third and fifth letters of the target. One-letter different primes were created by replacing the third letter with a consonant that did not occur elsewhere in the letter string, and two-letter different primes were created in a similar way by replacing both the third and fifth letters. Unrelated primes consisted of primes from the other conditions that shared no more than one letter with the target. The results of the empirical data and the simulation are shown in Table 4. As can be seen, the model provides a good qualitative fit to the data. Nonadjacent TL primes resulted in greater priming than did two-letter different primes but weaker priming than one-letter different primes.

In summary, the empirical data originally reported by Perea and Lupker (2004) provide critical information regarding the relative perceptual similarity of orthographic neighbors formed by replacing one or two letters or by transposing nonadjacent letters. As Davis (2006) noted, these data place rather strict constraints on theories of input coding by providing a yardstick by which to measure the differential impact of replacing a letter versus altering the position of that letter. A satisfactory model needs to capture the fact that *casino* and *caniso* are more similar to each other than are *casino* and *caviro* but less similar to each other than *casino* and

casiro. The spatial coding model succeeds in satisfying these dual constraints.

Simulation 9: Extreme Transpositions (Guerrera & Forster, 2008, Experiment 3). The TL priming effects discussed thus far indicate that not all of the letters of a word need to be in the correct position for that word to become activated by a letter string, and the displaced letters can be at least a couple of positions away from their correct position. One might therefore ask, what are the limits of TL priming? Guerrera and Forster (2008) sought to answer this question. To this end, they tested a range of anagram primes in which most or all of the letters were out of their correct position.

One condition that Guerrera and Forster (2008) tested in several experiments was one in which the exterior letters were maintained in their correct position but the six interior letters were each transposed with an adjacent letter (e.g., the prime for *SANDWICH* would be *snawdcih*); they refer to this as the T-I-6 prime condition. In each of their experiments, T-I-6 primes resulted in significant form priming, typically in the range of 20 ms to 30 ms. As Guerrera and Forster (2008, p.125) noted, "The degree to which the human word recognition system tolerates transposition in the input is quite remarkable."

In Experiment 3, Guerrera and Forster (2008) tested the limits of the system even further with a prime condition in which all eight letters of the target were transposed (e.g., the prime for *SANDWICH* would be *asdniwhc*); they refer to this as the T-all prime condition. In this case, there was no hint of a priming effect, and thus, Guerrera and Forster (2008) concluded that they had "now reached the limits of the system" (p. 133).

These two conditions provide important constraints on models of letter position coding. The strong priming for T-I-6 primes appears consistent with spatial coding, which predicts a match value of .76 for these primes. A much smaller match value (of .45) is computed for T-all primes. A simulation is required to establish whether the former prime gives rise to a facilitatory priming effect of the right general magnitude, and the latter prime results in no priming at all. This was the aim in Simulation 9, in which I tested three of the four prime conditions of Experiment 3 of Guerrera and Forster (2008). (The remaining prime condition is tested in Simulation 11, when string reversal is considered). The target words were the 96 eight-letter targets from Guerrera and Forster's (2008) experiment.

Table 4 shows the results of the simulation, together with the results from Guerrera and Forster's (2008) experiment. As can be seen, the T-I-6 prime condition showed a strong priming effect, in accord with the empirical data. By contrast, the T-all prime condition showed no priming effect in the data and a very small priming effect (of five cycles) in the model. The results of the simulation differ somewhat from theoretical predictions made by Guerrera and Forster (2008), who suggested that the moderately high match for T-all primes implied that the spatial coding model predicted "strong priming" (p. 137). As has been noted already, the relationship between match values and masked form priming effects is not straightforward, and Simulation 9 provides further evidence of this point.

Although the T-all priming effect in the model certainly could not be characterized as strong, there is some evidence of priming in this condition. Is this problematic for the model? Further empirical data are relevant to considering this question. Lupker and Davis (2009, Experiment 1A) replicated the T-all priming condition of Guerrera and Forster's (2008) experiment, using exactly the same stimuli. This experiment also showed a statistically nonsignificant priming effect, but there was a 9 ms advantage for T-all primes relative to control primes. The average of the priming effects observed in these two experiments is thus 4 ms, that is, quite close to the predicted priming effect of five cycles.

Simulation 10: Sandwich priming with extreme transpositions (Lupker & Davis, 2009, Experiment 1). The reason for the apparent disconnect between the match values for T-all primes and the priming effects observed in Simulation 9 is that, as in previous simulations, the prime activates lexical competitors of the target more strongly than the target itself. For example, consider a prime–target pair such as *baonmrla–ABNORMAL*. Though the prime has a match of .45 with the target, it is a better match for a number of other words, including *banner* (match = .625), *baron*, *baronial*, *formula*, *banana*, *bacteria,* and so on. The (moderate) activation of these word nodes interferes with the activation of the target word node, with the consequence that the prime provides no headstart to the processing of the target. By contrast, T-I-6 primes like *snawdcih* are virtually always better matches for the target (*SANDWICH*) than for any other words.

It may be apparent from the foregoing discussion that the sandwich priming technique offers a potential method for reducing the interference from such lexical competitors, thereby enabling T-all primes to become effective form primes. This possibility was tested by Lupker and Davis (2009). As noted above, using the same prime–target stimuli as Guerrera and Forster (2008), they replicated the latter's finding of a nonsignificant T-all priming effect when a standard masked priming methodology was used. However, when they used sandwich priming, they found a 40 ms T-all priming effect. With Simulation 10, I sought to simulate this finding. The simulation was identical to Simulation 9, except that sandwich priming was assumed; that is, the target was briefly presented prior to the prime of interest.

Table 4 shows the results of the simulation, as well as the relevant results from Lupker and Davis (2009). As can be seen, the switch to sandwich priming transformed the five cycle priming effect of Simulation 9 to a 29 cycle priming effect. The model's slight underestimation of the empirical effect may indicate that the σ parameter is underestimated for eight-letter words or that the model's account of the mechanisms underlying sandwich priming is incomplete; it may be noted that increasing the sandwich prime duration improves the quantitative fit to the data. Nevertheless, the more salient point is that in both the model and the data, sandwich priming has the effect of transforming a rather small T-all priming effect into a very large effect.

The finding that T-all primes can give rise to substantial form priming effects is important for two reasons. First, this finding demonstrates the extraordinary flexibility of the letter position coding system. Despite the fact that every single letter of the target has been displaced, these primes are sufficiently similar to the target to support its activation. This similarity is somewhat counterintuitive (the word *abnormal* does not leap out at one when confronted with its T-all prime *baonmrla*) and yet is exactly as predicted by the spatial coding model. The assumption of position-independent letter coding naturally leads to the prediction that even quite extreme anagrams are relatively similar to their base word (compared with all-letter-different controls). The fact that sandwich priming is a sufficiently sensitive methodology to detect

this similarity is the second noteworthy aspect of this finding. The great potential of the sandwich priming methodology constitutes powerful testimony to the theoretical and practical value of the computational model that led to its development.

C. Primes that involve letter string reversal. String manipulations like those considered in Simulations 6–10, in which a single pair of letters is transposed or in which there are a small number of pairwise transpositions, give rise to similar orthographic similarity scores in spatial coding and open-bigram coding models (e.g., Dehaene, Cohen, Sigman, & Vinckier, 2005; Grainger & van Heuven, 2003; Grainger & Whitney, 2004; Schoonbaert & Grainger, 2004; Whitney, 2001, 2004). However, the predictions of these models are distinguished by a special case of letter transposition that occurs when an entire string (or long substring) of letters is reversed, as in *draw* and *ward*. Such manipulations offer a particularly strong test of context-specific coding schemes in which letter position is coded by activating nodes that code where a letter occurs relative to another nearby letter: in particular, whether it occurs before or after this letter. For example, *draw* activates the open bigrams *DR*, *DA*, and *DW*, but not the reversed open bigrams *RD, AD*, or *WD* (which are activated by *ward*). Indeed, reversed letter strings like *draw* and *ward* do not share any common bigrams. By contrast, these letter strings are reasonably similar according to the spatial coding model.

Thus, examining effects of letter string reversal offers a means of testing fundamental assumptions of different models of letter position coding. Simulations 11 through 13 consider two different sorts of string reversal manipulations. In Simulations 11 and 12, the aim is to test whether the spatial coding model overestimates the similarity of reversed letter strings such that it predicts form priming in situations in which none is found (Davis & Lupker, 2010; Guerrera & Forster, 2008). Simulation 13 tests whether the spatial coding model can correctly predict form priming effects in situations where reversed-string priming is obtained (Davis & Lupker, 2010).

Simulation 11: Reversed-halves (RH) priming (Guerrera & Forster, 2008). RH anagrams constitute another form of prime–target relationship that was originally tested by Guerrera and Forster (2008). These anagrams are formed by reversing the letters in each half of a word (e.g., *DRAWBACK* => *wardkcab*). This manipulation greatly disrupts relative position but has a smaller disruptive effect on absolute position because the internal letters of each half are left adjacent to their original position (e.g., the *R* in *DRAWBACK* has shifted by only a single position in *wardkcab*). According to the spatial coding model, then, there should be a moderate similarity score for RH anagrams.

By contrast, open-bigram coding models predict that RH anagrams are relatively dissimilar. Indeed, given the standard restrictions on the distance separating the letters of an open bigram, RH anagrams like *DRAWBACK* and *wardkcab* do not share any open bigrams. This prediction about the perceptual similarity of RH anagrams is consistent with Guerrera and Forster's (2008) findings, which showed no evidence of a form priming effect when RH anagrams were used as primes.

The aim in Simulation 11 was to test whether the spatial coding model incorrectly predicts form priming for RH anagram primes. The simulation tested the same stimuli used by Guerrera and Forster (2008). Results are shown in Table 4, along with the corresponding results from Guerrera and Forster's experiment. The model predicts a priming effect of only three cycles for RH anagram primes, which is compatible with the null effect observed in the data.

Simulation 12: Reversed-interior (RI) priming (Davis & Lupker, 2009). One possible explanation for the lack of priming produced by RH anagram primes is that they differ from the target with respect to the exterior letters, which may play a particularly important role in lexical matching. This raises the possibility that a reversed-string prime could support priming if the exterior letters were preserved in their correct position. Davis and Lupker (2010) have recently investigated this possibility, constructing RI primes by reversing all of the internal letters of eight-letter words. For example, for the target *COMPUTER*, the RI prime was *cetupmor*. Control primes were formed by maintaining the exterior letters and replacing all of the interior letters with letters that do not occur in the target (e.g., *calibnar*). A standard masked priming paradigm produced the results shown in Table 4. The 5 ms difference between the RI condition and the control prime condition did not approach significance.

According to the spatial coding model, the predicted match for RI primes is reasonably high (.52, compared with .40 for control primes). This raises the possibility that the model incorrectly predicts form priming for RI anagram primes. The aim in Simulation 12 was to check this possibility. The model was tested with the same stimuli used by Davis and Lupker (2010). Results of the experiment and the simulation are shown in Table 4. As can be seen, the model and data agree in predicting virtually no priming for RI primes. In summary, although the spatial coding model predicts greater levels of similarity between reversed letter strings than other models, Simulations 11 and 12 show that the model correctly predicts the absence of masked form priming from the reversed letter string primes that have been examined empirically (Davis & Lupker, 2010; Guerrera & Forster, 2008).

Simulation 13: Sandwich simulations (Davis & Lupker, 2010). The previous discussion of T-all primes (Simulations 9 and 10) demonstrated that these primes do not give rise to priming in the standard masked priming paradigm but are associated with robust priming effects when the sandwich priming technique is used (Guerrera & Forster, 2008; Lupker & Davis, 2010). By analogy, Davis and Lupker (2010) speculated that reversed letter string primes that do not give rise to priming in the standard masked priming paradigm (e.g., Guerrera & Forster, 2008) might give rise to priming effects when the sandwich priming technique is used. That is, the absence of priming in the standard masked priming paradigm may reflect effects of lexical competition. For example, consider an RI prime–target pair such as *cetupmor–COMPUTER*. Though the prime has a match of .56 with the target, it has a slightly stronger match with a number of other words, including *tumour*, *stupor*, *camphor,* and *customer*. The (moderate) activation of these word nodes interferes with the activation of the target word node, with the consequence that the prime provides no headstart to the processing of the target. The sandwich priming technique has the potential to reduce the interference from such lexical competitors, so that reversed-string primes might become effective form primes.

The results obtained by Davis and Lupker (2010) are consistent with this conjecture (see Table 4). Using the same prime–target stimuli as Guerrera and Forster (2008), Davis and Lupker (2010) found a small but significant 9 ms priming effect for RH primes when sandwich priming was used. Likewise, they found a signif-

icant 23 ms priming effect for RH primes when sandwich priming was used. In Simulation 13, I sought to simulate these two sandwich priming experiments, using the same stimuli as in the experiments.

The results of the simulation are shown in Table 4. As can be seen, the switch to sandwich priming transformed the null RI priming effect of Simulation 12 to a 21 cycle priming effect, and increased the two-cycle reversed halves priming effect of Simulation 11 to a seven cycle priming effect. The magnitudes of these effects are close to the human data.

These findings provide a further demonstration of the usefulness of the sandwich priming technique. Moreover, they demonstrate that the null effects of reversed-string primes in standard masked priming effects cannot be interpreted as evidence for the weak (or zero) perceptual similarity of strings formed through string reversal manipulations. Rather, these findings suggest that such manipulations result in pairs of strings that are at least moderately similar to each other. This conclusion is consistent with spatial coding but is problematic for open-bigram coding models (this point is discussed further below).

D. Superset primes. Each of the string manipulations considered thus far (letter replacement, letter transposition, and string reversal) result in primes that preserve the string length of the target. In the next two subsections, I consider string manipulations that modify string length through the deletion or addition of letters. Letter insertion has the interesting property of modifying absolute letter position while maintaining relative order of letters (e.g., consider the order of the common letters in *special* and *specxyial*). This is not to say that letter insertion should have no impact on orthographic similarity. Although order is maintained, relative position information is modified slightly by letter insertion (e.g., the *c* in *specxyial* is further away from the letter *i* than the *c* in *special*). This disruption of letter contiguity leads to a slight reduction in the match value for each inserted letter. The empirical effects of letter insertion on masked priming have been investigated in a series of experiments reported by Grainger and colleagues (Van Assche & Grainger, 2006; Welvaert, Farioli, & Grainger, 2008). These experiments have tested the effect of number of inserted letters on form priming and have also examined whether letter insertion effects are dependent on the status of the inserted letter (repeated or unique). These two issues are considered in Simulations 14 and 15.

Simulation 14: Parametric manipulation of number of inserted letters (Van Assche & Grainger, 2006; Welvaert, Farioli, & Grainger, 2008). Van Assche and Grainger (2006) and Welvaert et al. (2008) reported several experiments investigating the effects of letter insertion on form priming. As Welvaert et al. (2008) noted, slightly different patterns can emerge from one experiment to another, so it is advisable to combine data from multiple experiments. To this end, they performed a meta-analysis based on seven experiments ($N = 248$) that included seven-letter word targets and primes that included between zero and three inserted letters (zero inserted letters corresponds to identity priming). This analysis revealed a graded effect of letter insertion, in which there was a cost of 11 ms per letter inserted; the linear regression equation explained 62% of the variance in priming effects.

The stimuli for Simulation 14 were constructed to parallel those used in the experiments of Van Assche and Grainger (2006) and Welvaert et al. (2008). The targets were a random set of 96 seven-letter words with no repeated letters. Primes were constructed by inserting zero, one, two, or three letters at either Position 4 (e.g., *abdomen*, *abdgomen*, *abdgcomen*, *abdgcxomen*) or Position 5 (*acquire*, *acquhire*, *acquhjire*, *acquhmjire*); in addition, a set of all-letter-different control primes was constructed.

As can be seen in Table 4, although the model underestimated priming by around six cycles for each condition, it was quite successful in capturing the linear relationship between number of inserted letters and priming effects. According to the model, this linear decrease in priming effects reflects the disruption of relative position information as further letters are inserted. The slight underestimation of priming may reflect the slight variation in masked priming methodology used in these experiments. Specifically, Van Assche and Grainger (2006) and Welvaert et al. (2008) used four stimulus fields rather than three, with the prime followed by a 16 ms backward mask that preceded the target stimulus; by contrast, the simulation used the same procedure as in other simulations; that is, there was no backward mask. The inclusion of the intermediate mask in the experiment allows extra processing time for the prime, which could increase masked priming effects.

Simulation 15: Superset priming (Van Assche & Grainger, 2006, Experiment 1). Having established that superset primes can produce robust form priming effects, Van Assche and Grainger (2006) proceeded to investigate whether the magnitude of these effects was affected by whether the inserted letter was a repeated letter (i.e., one that occurred already in the target) or a unique letter. They compared two repeated letter conditions: one in which the repeated letters were adjacent (e.g., *cabinet–CABINET* or *cabinet–CABINET*) and another in which the repeated letters were separated by two intervening letters (*cabinbet–CABINET* or *canbinet–CABINET*). There were two versions of the unique letter condition (e.g., *cabinxet–CABINET* and *caxbinet–CABINET*), to control for possible letter position effects. Finally, identity and unrelated prime conditions were included for comparison purposes. The simulation included each of these eight conditions and used the same targets as Simulation 14.

The match values computed by the spatial coding model are equivalent for the two repeated letter conditions and also for the unique letter condition (e.g., *cabinxet–CABINET*). This equivalence reflects the model's equivalent treatment of repeated and unique letters. As can be seen in Table 4, the equivalent match values across the repeat, repeat-displace, and unique (insert) letter conditions translate to approximately equivalent predicted priming effects. This pattern perfectly mirrors the pattern of the data.

It is worth noting that the equivalence of the displaced and adjacent repeated letter conditions is not a necessary prediction of all models. According to the discrete open-bigram coding scheme described by Schoonbaert and Grainger (2004), the displaced repeated letters should produce a better match than the adjacent repeated letters (.93 vs. .87). The same prediction follows from the current SERIOL model (Whitney, 2004), whereas the overlap open-bigram coding model predicts the opposite pattern (.90 vs. .96). Van Assche and Grainger (2006, p. 346) concluded that "in terms of the influence of letter repetition on relative-position priming effects, it appears that the SOLAR [i.e., spatial coding] model generates superior predictions relative to the other models examined."

E. Noncontiguous subset primes. Subset primes are formed by deleting letters of the target. It is appropriate to distinguish between two types of subset primes. The deletion of initial or final letters results in contiguous subset primes (e.g., *qual–quality*); these are considered in the next subsection. The deletion of internal letters of the target results in noncontiguous subset primes. Like superset primes, these primes modify absolute letter position while maintaining letter order (e.g., consider the order of the common letters in *special* and *spcal*). However, although order is maintained, the relative position is modified slightly by letter deletion (e.g., the *c* in *spcal* is closer to the letter *p* than the *c* in *special*). This disruption of letter contiguity results in a slight reduction in the match value, as does the absence of one or more of the letters of the template. The simulations reported in the present subsection examine the effect of number of deleted letters, the effect of deletions versus substitutions (Peressotti & Grainger, 1999), and the importance of relative letter position in letter deletion primes (Grainger et al., 2006; Peressotti & Grainger, 1995, 1999), as well as the importance of whether the deleted letter occurs only once or repeatedly in the target (Schoonbaert & Grainger, 2004).

Simulation 16: Deletion of repeated versus unique letters (Schoonbaert & Grainger, 2004, Experiment 1). Schoonbaert and Grainger (2004) tested subset priming effects for targets containing repeated letters. Their prediction (based on open-bigram coding) was that deleting a repeated letter (e.g., *trival–TRIVIAL*) would result in greater priming than deleting an unrepeated letter (e.g., *trivil–TRIVIAL*). Contrary to this prediction, primes formed by deleting repeated letters were not more effective than those formed by deleting unique letters. Schoonbaert and Grainger (2004) noted that this result is consistent with a spatial coding account because the spatial coding model treats repeated letters in the same way as unique letters and incorporates a mechanism to prevent letters doing "double-duty"; that is, a single letter cannot contribute to the overall match more than once (e.g., the *i* in *trival* cannot count toward both *i*s in *trivial*).

The English stimuli used in Simulation 16 were constructed in the same way as in Schoonbaert and Grainger's (2004; French) experiment. The critical targets were 64 seven-letter words containing repeated letters (where the repeated letters were not adjacent, and did not occur in initial or final positions). The primes were constructed by deleting either the second occurrence of the repeated letter, or an immediately adjacent letter. As in Schoonbaert and Grainger's (2004) experiment, possible letter position effects were controlled for by testing an equal number of control targets that contained no repeated letters and that were primed with letter deletion primes constructed in the same way as for the targets with repeated letters.

The results of the simulation are shown in Table 4. As in Simulation 14, the model's underestimation of the magnitude of priming effects by around five cycles across each condition may reflect the need to increase the prime duration in the simulation to simulate the use of the intervening backward mask in the experiment. Nevertheless, the overall pattern of the results demonstrates an excellent fit to the experimental data. Subset primes were extremely effective for both the repeated letter targets and the control targets, and the size of the priming effect did not appreciably differ as a function of whether the deleted letter was a repeated letter or a unique letter. These results, combined with those from Simulation 15, demonstrate that the method by which the model handles repeated letters is consistent with a broad range of priming data.

Simulation 17: Subset versus substitution priming (Peressotti & Grainger, 1999, Experiment 2). Peressotti and Grainger (1999) reported several experiments investigating various aspects of relative position priming. They sought to extend work previously reported by Humphreys, Evett, and Quinlan (1990), who found that the identification of target words could be primed by subsets that shared common letters with the target in the same relative position (e.g., exterior letters primed exterior letters and interior letters primed interior letters). For example, identification of targets like *BLACK* was facilitated both by substitution primes like *btvuk* (relative to control primes like *otvuf*) and by deletion primes like *bvk* (relative to control primes like *ovf*). Peressotti and Grainger (1999) attempted a replication of Humphreys et al.'s (1990) prime conditions, using the three-field masked priming technique and the lexical decision task.

Experiment 2 of Peressotti and Grainger (1999) manipulated prime relatedness and prime length to form the following four priming conditions: (a) the subset prime 1346 (e.g., *crtn–CARTON*), (b) the unrelated control prime *dddd* (e.g., *vsfx–CARTON*), (c) the substitution prime 1*d*34*d*6 (e.g., *czrtwn–CARTON*), and (d) the unrelated control prime *dddddd* (e.g., *vzsfwx–CARTON*). The description of primes in this and the following simulations adopts the common practice of using digits to indicate letter position, for example, the notation 1346 indicates a prime consisting of the letters from positions 1, 3, 4, and 6 of the target. The letter *d* (as in 1*d*34*d*6) indicates a letter that is not present in the target. Their results showed that subsets were effective form primes, relative to their controls, but that substitution primes were not (relative to their controls). Peressotti and Grainger (1999) concluded that the presence of nontarget letters in the substitution primes exerted an inhibitory effect on target identification.

Simulation 17 used stimuli that were constructed in the same way as Peressotti and Grainger (1999)'s (their experiment used French stimuli; hence, it was necessary to construct English stimuli with the same characteristics). There were eighty six-letter targets, each paired with four different primes. Table 4 shows the mean priming effects predicted by the model. As can be seen, both the subset primes and the substitution primes resulted in facilitatory priming, but the priming effect was larger for subset primes. The model slightly underestimates the 27 ms effect that Peressotti and Grainger (1999) found for subset primes in their Experiment 1, although these authors obtained smaller priming effects for this condition in a subsequent experiment in the same series (see below).

The magnitude of the predicted priming for substitution primes was very close to the observed effect. The latter effect was not statistically significant in Peressotti and Grainger's (1999) data, but other experiments suggest that there is a genuine priming effect to be found for two-letter different primes and six-letter targets (e.g., Perea & Lupker, 2004), as has also been observed for two-letter different primes and seven-letter targets (Lupker & Davis, 2009; see Simulation 4). Nevertheless, the critical aspect of the results is that both the model and the data show greater priming for subset primes than for substitution primes. This suggests that the mismatch inhibition produced by substituted letters outweighs

any beneficial effect these letters may have in helping to preserve relative position information for the remaining letters.

Simulation 18: The relative position priming effect (Peressotti & Grainger, 1999, Experiment 3). A second issue explored by Peressotti and Grainger (1999; again following on from earlier work reported by Humphreys et al., 1990, on the perceptual identification of primed targets) concerns the flexibility of letter position coding as revealed by subset priming. As discussed in detail already, there is considerable evidence for facilitatory priming from primes that transpose two adjacent letters of a target stimulus (e.g., *catron–CARTON*). One might therefore anticipate that subset primes that incorporate adjacent letter transposition would result in some form priming (e.g., that *ctrn–CARTON* would result in a priming effect not too much smaller than that for *crtn–CARTON*). However, this expectation is violated by the results of Peressotti and Grainger's (1999) Experiment 3 (see Table 4), which replicated the facilitatory effect of subset primes but showed no priming for subset primes in which the order of the two interior letters was transposed (i.e., 1436 primes like *ctrn–CARTON*) or in which the order of the two exterior letters was transposed (i.e., 6341 primes like *nrtc–CARTON*).

Simulation 18 had the same 80 word targets as Simulation 17. The 1346 and dddd prime conditions were also identical to those of Simulation 14, and the new conditions of 1436 and 6341 were included. Table 4 shows the mean priming effects predicted by the model. As can be seen, the 1346 subset primes showed a facilitatory priming effect of the same size as that effect observed by Peressotti and Grainger (1999); this correspondence suggests that the slight underestimation of subset priming observed in Simulation 14 is not problematic for the model. The model also correctly predicts a weaker priming effect for the 1436 subset primes, although there is evidence of some facilitatory priming (cf. Stinchcombe, Lupker, & Davis, 2010). Finally, the model predicts no priming for the 6341 primes, consistent with the nonsignificant priming effect reported by Peressotti and Grainger (1999). Thus, the model provides a good account of relative position priming.

F. Primes that involve letter string displacement. A key claim of the spatial coding model is that the recognition of letter strings is position invariant. The assumption of end-letter marking introduces a slight degree of position specificity (in the sense that the model computes larger match values when two strings share their initial and final letters), but it is nevertheless the case that the model predicts that a familiar letter string that ordinarily occurs in serial Positions 1–4 can be recognized even when it occurs in Positions 4 through 7. To date, researchers have examined such manipulations, in which letter strings are shifted rather than transposed or reversed, in very few masked priming experiments. These experiments have produced somewhat mixed results. On the one hand, Grainger et al. (2006) observed significant priming effects when substrings of the target word were shifted by as many as four letter positions. For example, in one experiment Grainger et al. (2006) found that priming nine-letter target words was equivalent for 12345 primes (e.g., *labyr–LABYRINTH*) and 56789 primes (e.g., *rinth–LABYRINTH*), despite the fact that the letters of the prime in the latter condition are shifted by four positions (forward) relative to the positions of the corresponding letters in the target. This result is consistent with position-invariant identification and poses a strong challenge to position-specific models.

On the other hand, Kinoshita and Norris (2008) have recently reported a masked priming experiment in which priming effects were not obtained for position-shifted primes. The targets in this experiment were eight-letter words and nonwords, and the critical primes were of the form 56781234; that is, the first half of the target was shifted forward by four positions, whereas the second half was shifted backward by four positions. This manipulation resulted in no priming, relative to an unrelated baseline condition. In Simulations 19 and 20, I investigate whether the spatial coding model can accommodate the dual (and potentially conflicting) constraints imposed by these two sets of results.

Simulation 19: Shifted-halves (Kinoshita & Norris, 2008). As noted above, Kinoshita and Norris (2008) have recently reported a masked priming experiment which, on the surface at least, appears to conflict with the findings of Grainger et al. (2006). As in the 56789 prime condition of the latter experiment, the primes consisted of substrings of the target that had been shifted from their normal position by four letter positions (although the targets were eight letter long rather than nine letters long). This substring was then concatenated with a substring corresponding to the initial four letters of the target. The resulting primes, of the form 56781234, showed no priming relative to an unrelated baseline condition.

From an empirical perspective, the absence of 56781234 priming is noteworthy. The results of Grainger et al. (2006) suggest that 5678 should be an effective prime for an eight-letter target, so it may be somewhat surprising that the addition of four further target letters should eliminate this effect. More important, from a theoretical perspective, the absence of 56781234 priming may appear to pose a strong challenge to position-invariant recognition. This challenge is especially profound in the case of models that attempt to achieve position-invariant recognition through matching of small sublexical chunks. For example, the overlap open-bigram model (Grainger et al., 2006) predicts a match of .85 between a target and a shifted-halves prime (e.g., *drenchil–CHILDREN*), and high match values are also produced by the discrete open-bigram and SERIOL models. These high values reflect the fact that *drenchil* and *children* share most of their local context, e.g., *c* is followed by *h*, *d* is followed by *r*, and so on.

However, from the perspective of the spatial coding model, Kinoshita and Norris's (2008) result is not so problematic because match values in this model are based on the whole string, and not just the similarity of the sublexical components. That is, *drenchil* and *children* share common substrings (1234 and 5678), but their whole-word match is not as good. The addition of the 1234 component will not increase the priming that can be generated by the 5678 substring because these additional letters are too far away from the expected position that is implied by the remaining letters. This point is most easily illustrated graphically. Figure 8 shows the signal-weight difference functions computed by the *children* word node when the input stimulus is *drenchil*. As can be seen, the node is in some sense sensitive to the overlap with both halves of the stimulus. The *d*, *r*, *e*, and *n* functions are perfectly aligned with each other, so that the node can "recognize" that the substring *dren* matches part of the template. Likewise, the *c*, *h*, *i*, and *l* functions are perfectly aligned with each other, so that the node can recognize that the substring *chil* matches part of the template. However, the signal-weight difference functions for these components are not aligned with each other (rather, they are quite distant), and so

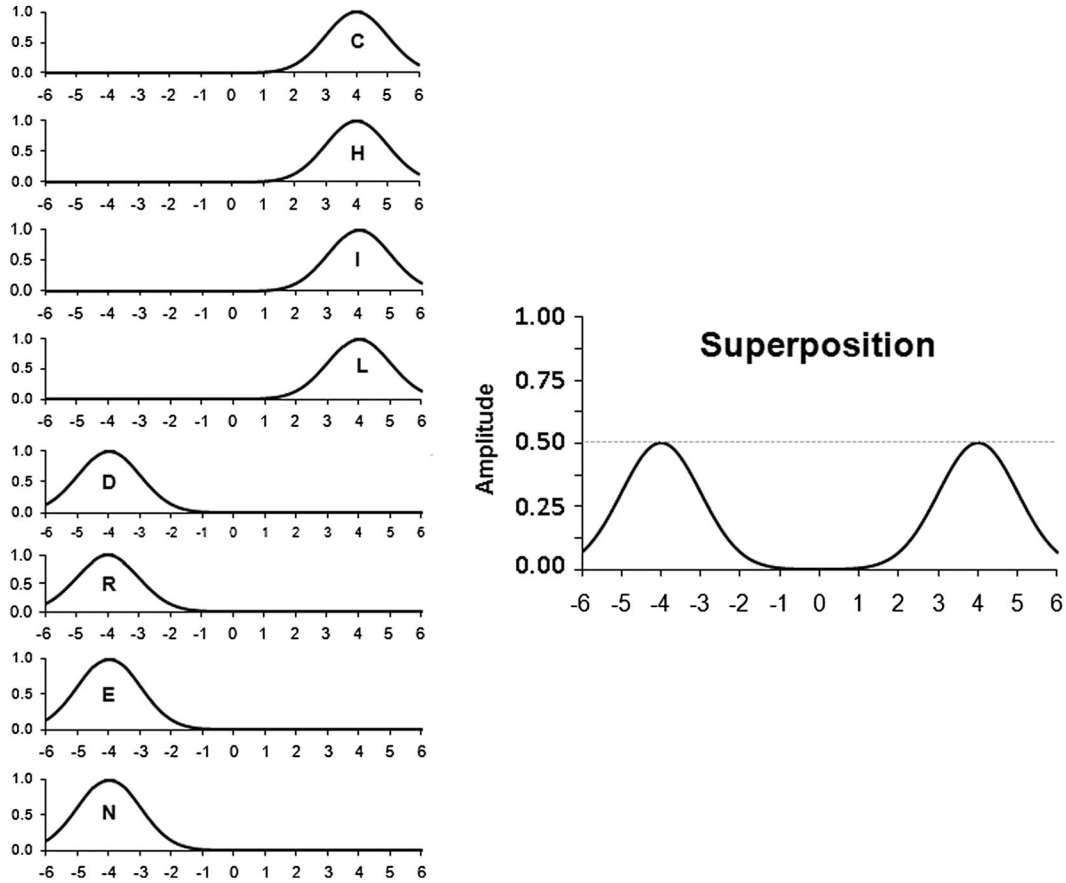

Figure 8. Superposition matching for shifted-halves stimuli (in this case, the input *drenchil* and the template CHILDREN).

there is not a complete (or even a particular close) match between the stimulus and the template. Effectively, when faced with a 56781234 prime, the word node must choose between two partial matches: one in which the stimulus has been shifted forward by four positions or one in which it has been shifted backward by four positions (the same stimulus cannot simultaneously have been shifted in both directions). Thus, with a spatial coding model, there is no reason to expect that a 56781234 prime will be more effective than a 5678 prime. As noted below, the same is not true of other current models.

Furthermore, the spatial coding model offers a couple of reasons to expect that a 56781234 prime will be less effective than a 5678 prime. The first reason relates to end-letter marking. Consider the target *INTERVAL*, and the primes *rval* versus *rvalinte*. In addition to the common substring *rval*, *rval* and *interval* share the property that their final letter is *l* (resulting in a match of $(4 + 1)/(8 + 2) = .5$). This is not the case for *rvalinte* and *interval*, which have different initial and final letters (resulting in a match of $(4 + 0)/(8 + 2) = .4$). Thus, the addition of the 1234 letters (i.e., *inte*) actually decreases the match. The second, more important reason to expect that a 56781234 prime will be less effective than a 5678 prime is that the additional 1234 letters will frequently result in a better match between the prime and other words. For example, *rvalinte* produces a closer match to eight-letter words like *reliance*, *relative*, *radiance*, *validate*, and so on, as well as to shorter words like *ravine*, *reliant*, and *recline* than it does to *interval*. Even though none of these matches is especially close, they may be sufficient to prevent *rvalinte* from functioning as an effective prime for the target *INTERVAL*.

In Simulation 19, I aimed to test this lexical competition account of the absence of masked priming for shifted-halves primes. The simulation had the same (eight-letter) primes and targets as Kinoshita and Norris (2009). As can be seen in Table 4, there was no indication of a difference between the shifted halves (56781234) prime condition and the control (all-letter-different) prime condition. This result is in accordance with the findings of Kinoshita and Norris (2009).

Simulation 20: Position-invariant priming (Grainger et al., 2006). The most dramatic examples of position-invariant priming presented to date come from Experiments 2 and 3 of Grainger et al. (2006). These experiments showed large subset priming effects for both initial subsets (e.g., 12345–1234567) and final subsets (34567–1234567). Although there was some indication of larger priming effects for initial-overlap subsets, the large priming effects for final-overlap subsets (37 ms for 34567 primes and 12 ms for 4567 primes) are difficult to reconcile with a position-specific letter coding model, even one that incorporates letter position uncertainty (e.g., Gomez et al., 2008).

In Simulation 20, I attempted to simulate these effects. The stimulus set for this simulation was formed by randomly selecting a set of 60 seven-letter word targets, subject to the constraint that targets contained no repeated letters. Each of these targets was paired with eight different primes, which were of the form (a) 12345, for example, *plast–PLASTIC*; (b) 34567, for example, *astic–PLASTIC*; (c) 13457, for example, *pastc–PLASTIC*; (d) *ddddd*, for example, *qmbtu–PLASTIC*, (e) 1234, for example, *plas–PLASTIC*; (f) 4567, for example, *stic–PLASTIC*; (g) 1357, for example, *patc–PLASTIC*; or (h) *dddd*, for example, *qmbt–PLASTIC*.

The results of this simulation show some similarities with the empirical data, as well as some differences. As in the data, there were (numerical) priming effects for all six related prime conditions. In particular, the model correctly predicts that priming should be obtained for both initial subsets (12345, 1234) and final subsets (34567, 4567); that is, priming was not specific to absolute serial position. The model also correctly predicted a similarly sized priming effect for noncontiguous subsets (13457, 1357). Overall, however, the predicted subset priming effects tended to be smaller in magnitude than the empirical effects, especially for initial overlap primes. I return to this discrepancy below. It is important to note, though, that the same model predicts priming for final subsets like 4567 (in this simulation) but not for shifted-halves primes like 56781234 (in Simulation 19). This pattern suggests that it is possible to resolve the apparent conflict between the results of Grainger et al. (2006) and Kinoshita and Norris (2008).

Summary of simulation results. Figure 9 plots the relation between the model's predicted priming effects and the empirically observed effects. The line of best fit (dashed line) has a slope of .98 and intercepts the *y*-axis at 1.76; the proximity to the origin and a slope of 1 indicates that the parameter choices were successful in achieving a good correspondence between the units of milliseconds in which the observed priming effects are measured and the units of processing cycles in which the predicted priming effects are measured. The plot illustrates the remarkably close fit between theory and data (r = .95, root-mean-square error = 5.72), across the full range of prime manipulations and observed priming effects (from −34 ms/−35 cycles up to +55 ms/+58 cycles). Overall, the model (assuming a fixed set of parameters) is able to account for 90% of the variance in 61 mean priming effects derived from a set of over 25 form priming experiments that span the entire range of letter string manipulations. In view of the typical variability associated with empirical priming effects, the obtained fit is likely to be close to the limits of observation.

Figure 9 also shows a few outliers, in which the model's predicted priming effect overestimates or underestimates the observed effect size (the most extreme outliers are highlighted in the third column of Table 5). The model overestimated neighbor priming effects for high frequency targets in Simulation 1; on the other hand, the model tended to underestimate neighbor priming effects for the low frequency targets in Simulations 1 and 2. It is possible that these outliers reflect spurious noise, but the model's account of neighbor priming as a function of target frequency may warrant closer scrutiny.

The most noteworthy outliers relate to the model's underestimation of priming effects for five-letter contiguous subset primes (Grainger et al., 2006) in Simulation 20. The observed priming effects are extremely large, considering that the prime omits two of the letters of the target. Indeed, the priming effect of 45 ms for 12345 primes is of a similar magnitude to the effect size that the model predicts for identity primes with these targets. The magnitude of the observed effects suggests that these subset primes may invoke processes that are beyond form priming. One possibility is that the first five letters of the target word is sufficient to provoke some form of expectancy (see Forster, 1998, for a discussion of this possibility). Although the model incorporates expectancy in the form of its top-down feedback, it may be that this feedback is insufficiently strong or that the implementation of feedback is incorrect. Another possibility is that the observed priming effects for contiguous subsets includes a morphological priming component, which would be outside the scope of the present model.

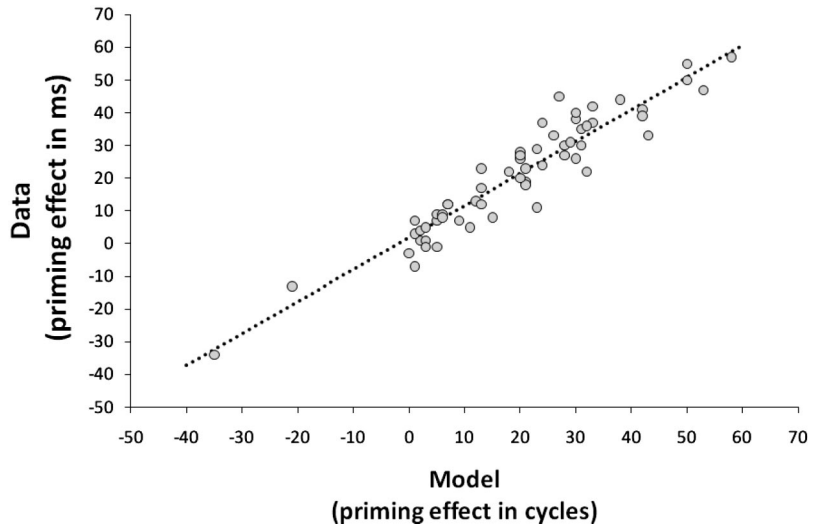

Figure 9. A plot of model predictions versus experimental data. Each point represents a priming effect modeled in the simulations.

Although the experiments modeled here focused on the issues of orthographic representations and processes, there was every opportunity for other types of representations and processes (i.e., phonological, morphological, or semantic) to influence the pattern of empirical results. Needless to say, the model would underestimate masked priming effects that are due to nonorthographic processes; for example, the present model would fail to predict associative priming effects such as those observed for prime–target pairs like *judge–COURT* (e.g., Perea & Gotor, 1997). Nevertheless, the fact that a purely orthographic model can provide such a good account of this large data set is of interest and appears to validate the general approach of seeking to study the workings of the orthographic lexicon in isolation. This approach clearly neglects important aspects of reading but nevertheless it appears that it can offer a fairly good characterization of early visual word identification processes.

Comparison of the Spatial Coding Model With Other Models

The spatial coding model differs in several ways from the original IA model, and thus it is not immediately apparent which of these differences might result in it offering a better account of the empirical database. In this section, I first demonstrate that the spatial coding model does indeed provide a better account of the data than does the IA model. I then consider a set of nine models, each of which differs from the spatial coding model with respect to one critical difference. Each model is tested on the same 20 simulations reported above, and their performance is compared with the model already tested. This approach makes it possible to assess the relative contribution of each of the differences of the spatial coding model from the original IA model. In the final part of this section, I turn to consideration of two alternatives to the standard approach that have been proposed in recent years: the overlap model (Gomez et al., 2008) and open-bigram coding models (e.g., Grainger & Whitney, 2004). Although both of these approaches are able to explain an impressive number of empirical observations, I conclude that neither is able to explain all of the results simulated here.

Original IA model. In order to conduct the same simulations that were reported above with the original IA model, a different parameterization of the spatial coding model was used, as described in the Appendix (where it is shown that this parameterization results in identical performance for the four-letter vocabulary used in the original IA model). Not surprisingly, the model failed to identify a large proportion of the stimuli, as the result of competition between words of different length, and this resulted in a rather poor correlation between the model's predicted priming effects and the empirical data ($r = .17$). To try to provide a fairer test of the model, I then ran each of the simulations using vocabularies of a fixed length that corresponded to the target words in that simulation. The model then succeeded in identifying the target stimuli, although it frequently made errors on the lexical decision task because its identifications were too slow. As a compromise, I set the n_{letter} parameter to .0, so that yes decisions could be made whenever the model succeeded in identifying the stimulus. This enabled the model to achieve a correlation between its predictions and the empirical data of .63 (see Table 5). Although the model was reasonably successful in predicting priming effects for letter substitution primes, it systematically underestimated priming effects for TL primes. For example, the model predicted no facilitatory priming for internal transposition primes in Simulation 21, compared with the 30 ms priming effect observed by Perea and Lupker (2003a). Likewise, the model systematically underestimated priming effects for other primes that disrupted absolute letter position, including inserted letter primes, deleted letter primes, reversed string primes (when sandwich primed), and primes involving letter string displacement. These difficulties are consistent with the problems with the original IA model that have been discussed elsewhere (e.g., Davis, 1999, 2006).

Next, I compare different parameterizations of the spatial coding model in order to assess the relative contribution made by different aspects of the model.

Model without position uncertainty. Most of the incorrect predictions of the original IA model stem from its lack of position uncertainty; for example, the above-mentioned underestimate of TL priming would presumably be corrected if the model incorporated some position uncertainty. One way to evaluate the importance of position uncertainty is to set σ equal to 0 in the spatial coding model, thereby eliminating letter position uncertainty. Predictably, the resulting model does rather poorly on the full set of 20 simulations (see Table 5). The correlation between model and data ($r = .73$) is still somewhat higher than the IA model, chiefly due to the model's superior predictions for identity priming and sandwich priming. Clearly, however, the assumption of position uncertainty is a prerequisite for explaining the empirical data.

A further question concerning position uncertainty relates to the spatial coding model's assumption that σ varies as a function of stimulus length. How critical is this assumption? To answer this question, I tested the model with a fixed value of σ ($\sigma_0 = 2$, $\kappa_\sigma = 0$); results are shown in Table 5. As can be seen, this model performs very well overall, with a correlation between theory and data of .93. However, this value of σ is too small for long stimuli. This is evident in the sandwich priming simulations with eight-letter stimuli, where the model greatly underestimates the priming for reversed-string primes. The problem is even more apparent in the simulation examining the effect of parametric letter insertion. Whereas the spatial coding model predicts priming effects close to those reported in Welvaert et al.'s (2008) meta-analysis, the model with a fixed σ predicts no priming at all for primes involving the insertion of three letters (row 41 in Tables 4 and 5). The reason is that the insertion of three medial letters causes the remaining letters on either side to be separated by too great a distance to mutually contribute to a lexical match (e.g., the *bal* and *cony* in *balxyzcony* are too far apart to both support the activation of the target *BALCONY*; compare the example shown in Figure 4C and the accompanying description). This does not cause a problem for the model with variable σ, as letter insertions also increase the degree of position uncertainty, so that the distance of three letters is not too far to enable the signal-weight differences from both halves of the word to contribute to the target match. In the model with fixed σ, a higher setting of σ results in a better fit to the letter insertion data but reduces the goodness of the fit to other experiments (e.g., the model predicts too much priming for T-all primes in Simulation 9 and too small a difference between neighbor and neighbor-once-removed primes in Simulation 7). Thus, the qualitative fit to the data is superior if σ is assumed to vary with stimulus length.

Table 5
Summary of the Results of the 20 Simulations for the Spatial Coding Model and 10 Other Models

Index	Data	SCM	IAM	σ = 0	σ = 2.0	No PI	No MMI	No ELM	No MF	No LMI	IA decay	No TDF
1	**−34**	−35	−33	−36	−35	−35	−32	**−49**	−32	−32	**−49**	−35
2	**26**	20	29	23	19	21	21	**11**	16	16	25	20
3	**−13**	−21	−13	−21	−21	−21	−20	**−27**	−21	−19	−24	−21
4	**11**	**23**	**31**	**25**	**22**	**23**	**23**	17	20	21	**25**	**23**
5	**12**	7	**1**	8	7	8	7	4	4	6	9	7
6	**28**	20	26	20	19	20	21	**13**	**17**	19	22	20
7	**1**	2	9	3	2	2	−1	−1	−3	1	1	2
8	**−3**	0	5	2	−1	0	2	1	−4	−1	−1	0
9	**3**	1	11	2	1	1	0	0	−1	0	1	1
10	**22**	32	24	32	32	**33**	31	29	30	**34**	33	32
11	**17**	13	16	14	13	15	18	**5**	**6**	12	14	13
12	**1**	3	7	3	3	4	8	3	1	3	3	3
13	**4**	2	3	3	2	2	2	2	2	2	2	2
14	**−7**	1	2	1	1	1	0	1	1	1	1	1
15	**57**	58	**29**	58	58	58	59	55	58	59	**51**	58
16	**33**	43	23	43	43	43	**48**	38	43	**45**	36	43
17	**27**	28	**14**	28	27	28	**39**	20	27	29	23	27
18	**8**	15	6	15	15	15	**31**	6	16	16	13	15
19	**7**	5	4	5	5	5	**20**	1	5	5	7	5
20	**47**	53	**35**	53	53	53	51	51	52	51	58	53
21	**30**	31	**−2**	**0**	35	31	22	23	31	31	32	32
22	**13**	12	**1**	**−1**	15	12	7	22	6	6	15	12
23	**30**	28	27	29	28	29	29	20	24	28	33	28
24	**31**	29	27	29	29	29	30	20	26	29	33	29
25	**19**	21	12	**6**	22	21	19	10	15	19	26	21
26	**18**	21	12	**6**	23	21	19	10	16	20	26	21
27	**38**	30	**27**	31	30	31	30	**24**	**26**	29	33	30
28	**24**	24	16	**8**	25	25	15	15	21	23	28	24
29	**7**	9	17	11	9	10	14	4	5	5	13	9
30	**26**	30	**4**	**2**	25	31	25	29	31	31	33	31
31	**−1**	5	2	2	4	10	2	**13**	5	3	**9**	5
32	**9**	5	2	2	4	10	2	13	5	3	9	5
33	**40**	30	**−1**	**4**	**26**	35	**28**	42	30	30	**28**	30
34	**−1**	3	2	1	3	4	2	4	3	2	5	3
35	**5**	3	0	0	1	3	0	2	2	2	3	3
36	**9**	6	−1	1	4	6	8	14	6	5	15	6
37	**23**	21	**0**	**0**	**11**	22	27	**10**	22	22	**7**	21
38	**55**	50	**35**	50	50	50	46	49	49	49	50	50
39	**44**	38	**3**	**2**	35	37	**32**	35	37	40	39	38
40	**33**	26	**3**	**2**	**18**	23	**19**	**18**	**23**	26	28	26
41	**22**	18		**2**	**2**	15	**9**	**7**	**12**	17	21	18
42	**50**	50	**35**	50	50	50	46	49	49	49	50	50
43	**41**	42	**7**	**10**	39	38	36	39	41	44	42	42
44	**39**	42	**7**	**10**	39	36	36	39	41	38	42	42
45	**39**	42	**7**	**10**	40	37	37	40	42	44	43	42
46	**42**	33	**18**	**16**	33	34	**27**	**28**	**30**	34	34	30
47	**35**	31	**18**	**19**	31	32	**25**	30	29	32	32	28
48	**37**	33	**15**	**14**	33	34	**26**	28	29	32	35	29
49	**36**	32	**16**	**18**	31	33	**25**	**25**	29	31	32	29
50	**27**	21	**2**	**2**	22	**14**	**14**	**10**	20	21	23	**15**
51	**12**	13	15	13	13	13	18	4	14	13	15	13
52	**20**	21	**2**	**2**	22	14	14	10	20	21	13	15
53	**5**	12	3	2	13	13	1	2	10	8	15	6
54	**7**	1	0	1	1	1	0	1	1	1	1	1
55	**7**	1	0	1	1	−1	0	−1	0	0	1	1
56	**45**	**28**	**21**	**28**	**28**	**29**	**21**	**35**	**26**	**23**	**28**	**13**
57	**37**	**27**	**0**	**27**	**27**	**2**	**19**	**29**	**21**	**23**	**28**	**11**
58	**29**	23	**1**	**3**	23	**17**	**17**	**19**	24	25	26	**18**
59	**23**	13	18	13	13	15	**2**	21	**5**	**7**	20	**5**
60	**12**	8	**0**	8	8	**1**	**1**	12	**2**	4	11	4

Table 5 *(continued)*

Index	Data	SCM	IAM	$\sigma = 0$	$\sigma = 2.0$	No PI	No MMI	No ELM	No MF	No LMI	IA decay	No TDF
61	**8**	6	1	1	5	3	1	9	9	5	13	5
r		0.950	0.632	0.728	0.931	0.912	0.870	0.918	0.931	0.937	0.937	0.906

Note. Index refers to indices shown in Table 4. The column labeled Data shows the observed priming effects. Values in the column labeled Data are in bold because all of the (model) values in the other columns are intended to be compared to these (data) values. Cases in which there is a large discrepancy between the model value and the corresponding data value are in bold, to highlight data points where the various models appear not to provide a good fit to the data. SCM = spatial coding model; IAM = the original interactive activation model; $\sigma = 0$ refers to the model with no letter position uncertainty; $\sigma = 2.0$ is the model that assumes a fixed value of σ for all stimulus lengths; No PI is the model without position invariance; No MMI is the model without mismatch inhibition; No ELM is the model without end-letter marking; No TDF is the model without top-down feedback; No LMI is the model without length-mismatch inhibition; No MF is the model without the masking field parameter; and IA decay is the model that uses the original IA-style node decay equation and parameters.

Model without position invariance. A critical difference between the spatial coding model and the standard approach to orthographic input coding is the assumption of position-invariant coding in spatial coding as compared with the position-specific coding assumed in the standard approach. The formulation of the equations underlying the spatial coding model enables the importance of this difference to be tested by varying a single parameter (see Equation A1 in Appendix). This parameter change effectively transforms the position-invariant spatial coding model into a position-specific coding model. Simulation results for this model are shown in Table 5. Overall, the model performs quite well, with a correlation between model and data of .91. Indeed, in many cases, this model makes identical predictions to those of the spatial coding model. However, there are a number of priming effects that are greatly underestimated by the position-specific model. Each of these priming effects involves subset priming. The most striking differences are for the final-overlap subset primes (e.g., *lcony–BALCONY* and *cony–BALCONY*) tested in Grainger et al. (2006). The data show priming effects of 37 ms and 12 ms for these two conditions. Although the position-invariant spatial coding model underestimates priming in the first case, it nevertheless predicts substantial priming in both of these conditions (of 24 cycles and seven cycles, respectively). By contrast, the position-specific model predicts no priming for either of these conditions (predicted effects of two cycles and one cycle). This failure to predict position-invariant priming is exactly as would be expected for the position-specific coding model. Although the set of experiments simulated here includes only one experiment that illustrates position-invariant priming, the same phenomenon was shown repeatedly in four separate experiments reported by Grainger et al. (2006). Position-invariant priming has also been observed in several morphological priming experiments (e.g., Crepaldi, Rastle, Davis, & Lupker, 2010; Duñabeitia, Laka, Perea, & Carreiras, 2009).

The other priming effects that are greatly underestimated by the position-specific model are the noncontiguous subset primes 1346 and 13457, tested by Peressotti and Grainger (1999) and Grainger et al. (2006), respectively. Although these primes share their first letter with the target, the deletion of the second letter means that the absolute position match is greatly disrupted. In the position-invariant spatial coding model, the resonating difference for these primes is -1; that is, the model is sensitive to the fact that the best relative position match is observed by the letters that occur one position earlier in the prime than in the target. The position-specific coding model cannot capture this aspect of the similarity between the prime and the target, and hence, it systematically underestimates priming for noncontiguous subset primes like 1*d*34*d*6 and 1*d*345*d*7. Thus, despite the model's quite good quantitative fit with the full data set, the failure of the position-specific coding model to explain priming for final overlap and noncontiguous subset primes is a critical flaw.

Model without mismatch inhibition. Another important respect in which the spatial coding model differs from the standard approach to orthographic input coding is its replacement of letter-word inhibition with nonspecific mismatch inhibition. To investigate the importance of this mismatch mechanism, the same set of 20 simulations was tested with a model in which the γ_{LW} parameter was set to 0; that is, mismatch inhibition was switched off. This model performed fairly well overall, with a correlation between theory and data of .87 (see Table 5). However, there are a few phenomena where this model does worse than the model with mismatch inhibition. One of these phenomena, perhaps surprisingly, is TL priming, where the model without mismatch inhibition predicts smaller priming effects than observed in the data (e.g., see rows 21 and 28). The match between theory and data could be improved by increasing σ. The more interesting aspect of this comparison, though, is that it reveals that one component of the TL priming effect seen in Simulations 6 and 8 is based on the fact that the TLs are not incompatible with the target, unlike the replacement letters in the orthographic control.

Although the model without mismatch inhibition can readily be adjusted to provide a good account of TL priming, there are two other phenomena that may not be so easy to accommodate. The first is the sandwich priming effects for primes that differ from the target by several letters. As can be seen in rows 16 through 19, the model without mismatch inhibition greatly overestimates the magnitude of priming for primes that differ from the target by 2, 3, 4, or 5 letters. By contrast, the model with mismatch inhibition is able to provide a good account of the observed priming effects, by virtue of the fact that increases in the number of replaced letters lead to increased levels of inhibition to the target node.

The other critical challenge for the model without mismatch inhibition is how to explain Peressotti and Grainger's (1999) finding that subset primes (1346) are (numerically, at least) more effective primes than double replacement primes (1*d*34*d*6). This model predicts a small difference in the opposite direction because the only effect of the replacement letters on the match value is a positive one: These letters ensure that the remaining letters (1346)

are in the correct positions relative to each other (i.e., letter contiguity is preserved over the prime and target stimuli in 1*d*34*d*6–123456 but not in 1346–123456). By contrast, the model with mismatch inhibition correctly predicts greater priming for 1346 than for 1*d*34*d*6, as a consequence of the inhibition that the two mismatching letters contribute to the target in the latter case. The difference between the 1346 and 1*d*34*d*6 priming effects was not statistically significant in Peressotti and Grainger's (1999) experiment, and thus further investigation of this difference would be desirable. Nevertheless, initial indications are that the mechanism of mismatch inhibition may play a critical role in explaining orthographic similarity data.

Model without end-letter marking. A final important respect in which the spatial coding model differs from the standard approach is in its introduction of end-letter marking. To investigate the importance of this aspect of the model, I tested an alternative model in which the weights from the two exterior letter banks were zeroed. Once again, this model performed quite well with respect to its overall fit to the data ($r = .92$, see Table 5). However, closer inspection of the results reveals that this model overestimates priming effects for primes that differ from targets with respect to their exterior letters. For example, the model fails to capture the difference in the magnitude of priming for TL-final primes relative to internal transpositions (Perea & Lupker, 2003a). Likewise, the model overestimates priming for the T-all and RH primes of Guerrera and Forster (2008; see also Lupker & Davis, 2009). Furthermore, the model without end-letter marking underestimates priming effects for primes and targets that shared the same exterior letters, as in the reversed interior primes of Davis and Lupker (2009), the deleted letter primes of Peressotti and Grainger (1999) and Schoonbaert and Grainger (2004), the inserted letter primes of Van Assche and Grainger (2006) and Welvaert et al. (2008), and the displaced letter string primes of Grainger et al. (2006). Although these results need not imply that the specific mechanism of dynamic end-letter marking is the correct way to capture the special status of exterior letters, it seems clear that some general mechanism of this sort is required to account for the empirical database on masked form priming effects.

Model without masking field parameters. The spatial coding model includes two mechanisms designed to facilitate competition between words of different lengths: masking field interactions and length-mismatch inhibition. According to masking field principles, nodes that code longer words have a competitive advantage over nodes that code shorter words (e.g., Grossberg, 1978). The magnitude of this advantage depends on the parameter mf. To examine the impact of masking field principles on the model's performance, I tested the 20 simulations with a model in which mf was set at 0. This model performed very similarly to the masking field model ($r = .98$), but the latter model typically resulted in a slightly closer fit to the data (the correlation between model and data for the nonmasking field model was .93). One difference between the two models that is not apparent from the lexical decision latencies was that the nonmasking field model occasionally made identification errors in which a low-frequency target word was misidentified as a high-frequency shorter word (e.g., on the trial *wenve–WEAVE*, the target was misidentified as "we"). A related phenomenon is the model's underestimation of priming effects for subset primes that are several letters shorter than the target, particularly if the prime is itself a word (e.g., *fort–FORTUNE*; see rows 59 and 60). This reflects the fact that in the nonmasking field model, nodes that code seven-letter words have greater difficulty in suppressing shorter words.

Model without length-mismatch inhibition. Similar comments apply to the model's assumption of length-mismatch inhibition. A model with no length-mismatch inhibition (i.e., $\gamma_{len} = 0$; see Table 5) performed very similarly to the model with length-mismatch inhibition, with the exception that it did slightly worse with respect to its predictions for the contiguous subset primes in Grainger et al. (2006). In summary, the assumptions of masking field principles and length-mismatch inhibition are helpful for the network's lexical selection mechanism, in that they help the best-matching lexical candidate to inhibit subset and superset competitors. However, neither of these assumptions is critical for explaining the orthographic similarity data simulated here.

Model without match-dependent decay. The spatial coding model incorporates a slightly different form of activity decay than does the original IA model. Specifically, it is assumed that the rate of decay is modulated by the match between the bottom-up input and the template. This assumption (in slightly different form) was originally proposed by Lupker and Davis (2009) to simulate sandwich priming effects. They observed that the exponential decay assumed by the original IA model caused the activity triggered by the initial sandwich prime to dissipate quite rapidly, making it difficult to account for the magnitude of sandwich priming effects. The assumption of match-dependent decay implies that a node that has been activated by the initial sandwich prime can maintain its activation if the critical prime is similar to the template but will decay rapidly for primes that are dissimilar to the template (as will be the case for control primes).

To test the impact of this modification to the model, the original form of decay was simulated by setting the DecayCutoff parameter equal to 1 and the DecayRate parameter equal to .07; in addition, the FreqBias parameter was set equal to 1.0 (otherwise the model often responds no to low frequency words). As can be seen in Table 5, the resulting model provides quite a good account of the data ($r = .94$). Future modeling is required to determine whether the proposed match-dependent decay has any advantage over the original form of activity-dependent decay.

Model without top-down feedback. The final variant of the model that I consider is one in which there is no top-down feedback (i.e., $\alpha_{WL} = 0$). The performance of this model is virtually identical to the model with top-down feedback, with the exception of the models' predictions for the contiguous subset primes of Grainger et al. (2006). Here, the model without top-down feedback predicts priming effects that are typically around 10 cycles smaller than the model with top-down feedback. Given that the latter model is already underestimating the magnitude of the observed effects, this difference implies that the model without top-down feedback greatly underestimates contiguous subset priming effects.

On the one hand, the above result could be interpreted as strong support for the role of top-down feedback in visual word identification. Indeed, perhaps the spatial coding model could do an even better job of fitting the data if its top-down feedback was greater or implemented slightly differently. If top-down feedback is very strong (e.g., $\alpha_{WL} = 10$), the model predicts considerably stronger priming effects for the subset primes of Grainger et al. (2006), thereby reducing (although not altogether eliminating) the

underestimation of priming for these data points. The reason for this is that top-down feedback causes the model to "fill in" the missing letters of the target, for example, the prime *balco* activates the *BALCONY* word node, which in turn activates the "missing" letters *N* and *Y* at the end of the stimulus.

On the other hand, there are grounds for being somewhat cautious regarding the role of top-down feedback. Although the model with strong top-down feedback does a better job of fitting the Grainger et al. (2006) data, the means by which it achieves this better performance is directly activating the missing letters (e.g., the stimulus *bayon* leads to the activation of the *bayonet* word node, which in turn leads to the activation of the final letters *e* and *t*). This is exactly the form of "hallucination" that Norris, McQueen, & Cutler (2000) cite as a reason not to include top-down feedback in models of perception. The problem is not that the model strongly activates a superset of the stimulus—it is that top-down feedback overwrites the trace left by the stimulus, such that the veridical record of the input is replaced by an expectation. The system then has no means of recovering from its error, that is, the nonword *bayon* is liable to be consistently read as *bayonet*. It may be that a more appropriate form of top-down feedback mechanism is possible. For example, top-down feedback could influence receiver nodes rather than letter nodes, so that the model retains some trace of the original input against which categorizations can be verified. This is an area for future investigation, and the study of subset priming may be fertile territory for the continuing debate between interactive and noninteractive models of perception (e.g., Bowers & Davis, 2004; McClelland, Mirman, & Holt, 2006; Norris et al., 2000).

Comparison of the model with other alternatives to the standard approach. The foregoing discussion has focused on the respects in which the spatial coding model improves on the standard approach, as exemplified by the IA model and other well-known computational models of visual word recognition. However, the spatial coding model is not the only alternative to the standard approach, and to conclude this section I directly compare the spatial coding model with other, newer models, specifically, (a) the overlap model (Gomez et al., 2008) and (b) the open-bigram models (e.g., Grainger & van Heuven, 2003; Whitney, 2001).

The overlap model. The overlap model (Gomez et al., 2008) is a noisy version of position-specific coding, in which the representation of a letter extends into adjacent positions. The model assumes a separate position-uncertainty Gaussian-distribution function for each letter of the stimulus and each letter of the template. Gomez et al. (2008) showed that the model could provide a very good fit to forced-choice perceptual identification data.

The overlap model has much in common with the spatial coding model. Both models assume that letters (rather than letter pairs or triples, for example) are the fundamental perceptual units in the matching process, and both models assume letter-position-uncertainty functions. The models vary with respect to the number of parameters they use to model position uncertainty—the spatial coding model uses only a single position uncertainty parameter, whereas the overlap model assumes one parameter for each letter position, to capture variations in letter position uncertainty across the stimulus (in particular, the first letter is associated with a much narrower uncertainty distribution than other letters, a characteristic that the spatial coding model captures through the use of end-letter marking). Nevertheless, Gomez et al. (2008) showed that a simplified version of the overlap model that assumes only two position-uncertainty parameters was also able to achieve a good fit to forced-choice perceptual identification data.

The key difference between the overlap model and the spatial coding model is the assumption of position-specific letter representations versus position-invariant letter representations, respectively. This difference is critical when the stimulus and the template overlap at different absolute positions, as in examples like *wildcat* and *cat*. It would be impossible for the overlap model to detect the *cat* in *wildcat*, whereas the position-invariant recognition of the spatial coding model makes it straightforward to detect embedded words like this. Another set of priming effects that are likely to be underestimated by the overlap model are those involving noncontiguous subset primes, as in the experiments reported by Peressotti and Grainger (1999) and Grainger et al. (2006).

One way to think about the spatial coding model is as a sliding overlap model. That is, the superposition matching algorithm implements a version of the overlap model in which the overlapping Gaussian functions representing the input are allowed to slide laterally across those representing the template until the maximum match is obtained. For example, in the *wildcat* example, the *CAT* word node computes three signal-weight differences of four. The peak in the superposition function at this point can be interpreted as reflecting the idea that the overlapping Gaussian functions representing the input have shifted four positions across the spatial code for the word *CAT* to "find" the point at which there is a maximum (in this case, perfect) overlap between the stimulus and the template. In the case of inputs that do not require any shifting to find the maximal overlap, the two models make quite similar predictions. Indeed, the spatial coding model could fit data such as those collected by Gomez et al. (2008) at least as well as the overlap model if it were given the same number of parameters (i.e., separate values of σ for each letter position). However, in the case of inputs that require shifting to find the maximal overlap (e.g., the masked form primes tested by Grainger et al., 2006), the spatial coding model can capture data that are beyond the scope of the overlap model.

The position specificity of the overlap model must be regarded as a critical weakness of this approach, not simply from the perspective of explaining form priming data but, more important, from the perspective of explaining morphological processing. Skilled readers are (indeed, must be) able to recognize the commonality of the morpheme *cat* across familiar words like *cat*, *catburglar*, and *wildcat*, as well as unfamiliar forms like *cathole*, *blackcatday*, and *supercat*. There are strong grounds for rejecting any approach to coding letter position that is fundamentally unable to support such position-invariant recognition.

A theoretical approach to orthographic input coding that is very similar to the overlap model has been adopted in the Bayesian reader model (Norris, 2006; Norris & Kinoshita, 2008). The Bayesian reader is a stimulus sampling model that assumes that readers are (approximately) optimal Bayesian decision makers. Unlike the overlap model, this model incorporates a lexical selection mechanism and has been extended to simulate masked priming (Norris & Kinoshita, 2008). The model integrates evidence over time from the prime and the target so as to make an optimal decision (i.e., in the case of lexical decision, is the stimulus a word?). Thus, the Bayesian reader model is not restricted to making predictions about match values and is capable of simulating masked priming (though to date the only masked priming

result from the lexical decision task that the model has been shown to simulate is a null effect of prime congruency observed by Norris & Kinoshita, 2008).

The lexical selection mechanism of the Bayesian reader is quite different from the lexical competition process of the IA, SOLAR, and related competitive network models. One important consequence of this difference is that the Bayesian reader cannot readily accommodate lexical competition effects in masked priming (cf. Bowers, in press) and thus is unable to account for the inhibitory priming effects of Davis and Lupker (2006) that were modeled in Simulation 1. Furthermore, the absence of lexical competition within the model makes it difficult to see how the Bayesian reader could accommodate other masked form priming phenomena that bear the hallmarks of lexical competition, such as the shared neighborhood effect (van Heuven et al., 2001), the multiple-letter replacement constraint (e.g., Schoonbaert & Grainger, 2004), and sandwich priming effects (Davis & Lupker, 2009; Lupker & Davis, 2009). (See Simulations 2, 3, 4, 5, 10, and 13 of the present article.) Thus, it seems likely that the present version of the Bayesian reader faces considerable challenges, although simulations of the model are required to properly evaluate its fit to the data.

Open-bigram models. A popular approach in recent attempts to solve the problem of letter position coding involves the assumption of open bigrams (e.g., Dehaene et al., 2005; Grainger & van Heuven, 2003; Grainger & Whitney, 2004; Schoonbaert & Grainger, 2004; Whitney, 2001, 2004). An open bigram refers to an ordered pair of letters that is not necessarily contiguous in the input stimulus; for example, the word *cat* includes the open bigram *ct*. A number of different versions of open-bigram coding have been proposed; these versions vary with respect to (a) their sensitivity to letter contiguity, (b) the maximal distance between the letters in an open bigram, and (c) their coding of exterior letters. I focus here on the general characteristics of open-bigram coding models (for detailed descriptions of the different versions, see Davis & Bowers, 2006; Grainger & van Heuven, 2003; Whitney, 2004).

There are some similarities between open-bigram coding models and spatial coding. Both approaches result in relatively flexible letter position coding, enabling the explanation of phenomena such as TL similarity. Furthermore, both models can support (approximately) position-invariant identification. For example, open-bigram models can explain the capacity to detect *cat* in *wildcat* because both letter strings contain the open bigrams *ca*, *ct*, and *at*.

An obvious difference between open-bigram coding models and spatial coding is the fundamental unit of matching. One consequence of this difference that is worth noting relates to the nature of the matching process. In spatial coding, the letters of the input stimulus are directly matched against the whole template, whereas in open-bigram coding, the letters of the input stimulus are matched against open bigrams, and then these units are matched against the template. It follows that open-bigram coding can match multiple different subcomponents of a word even when these subcomponents are positioned differently, relative to each other, in the stimulus and the template. For example, when open-bigram coding is assumed, the stimulus *pondfish* is an excellent match for the word *fishpond* because the components *fish* and *pond* maintain their local context (e.g., the bigrams *fi*, *fs*, and *fh* are present in both *fishpond* and *pondfish*). This prediction about orthographic similarity runs into problems when attempting to explain both masked priming data (Kinoshita & Norris, 2008; cf. Simulation 19) and data from unprimed lexical decision (Crepaldi et al., 2010). Spatial coding, by contrast, handles these data well because the stimulus *pondfish* is a relatively poor match for the word *fishpond*: for example, in terms of the sliding overlap idea discussed earlier in this section, *pondfish* must be shifted four positions to the right to find the overlap in the word *fish*, but this shifts the constituent *pond* even further from its correct position.

Another key difference between spatial coding and open-bigram coding concerns the directionality inherent in the latter model. As noted, open bigrams are *ordered* letter pairs, and these units are not activated when the order of the letters is reversed. For example, the anagrams *ward* and *draw* do not share any open bigrams. Recently, in a number of experiments researchers have sought to test the strong predictions about the effects of letter string reversal that follow from open-bigram coding (Davis & Lupker, 2009; Whitney & Grainger, 2008; see Simulations 11 through 13). These experiments indicate that when a suitably sensitive methodology is used, the perceptual similarity of pairs of reversed letter strings is clearly apparent. Such data pose critical problems for open-bigram coding.

In addition to the empirical problems with open-bigram coding, there are also some important theoretical objections to this method of coding letter position (I focus here on general problems with open-bigram coding; for a discussion of problems with a specific open-bigram model, the SERIOL model, see Davis, in press). One objection relates to the plausibility of the notion of open bigrams that entail the visual system discarding intermediate letters. Another objection concerns the lack of generality of the open-bigram solution to encoding position, given that this solution is not helpful for converting spelling to sound. In view of the fact that a different type of letter position code is required to accomplish this mapping (one that is capable of encoding position-invariant relationships between graphemes and phonemes), it is not clear what is gained by assuming a less versatile position code for lexical matching. This lack of generality also extends to other aspects of position coding. Although it has not generally been recognized (though see Davis, 1999), the problem of coding the relative order of morphemes gives rise to more or less the same issues as coding letter position. Here, however, an open-bigram type solution seems unsatisfactory. Representing all possible letter pairs in English requires 26×26 open bigrams, a number that does not seem unfeasible for representing letter order. By contrast, representing all possible morpheme pairs would require a highly implausible number of units. Rather, the ability to encode novel compounds, as well as to distinguish reversible compounds like *overtake* and *takeover*, must rely on a capacity to dynamically assign order information to individual constituents, as in the spatial coding scheme. In summary, although open-bigram coding models make many similar predictions to spatial coding, the open-bigram approach faces some critical empirical and theoretical challenges.

Summary and Conclusions

This articles has described the spatial coding model of letter position coding and lexical matching. I have previously argued (Davis, 2006; Davis & Bowers, 2006) that this is the only existing model that can satisfactorily account for critical phenomena related to ortho-

graphic input coding and lexical matching. I have further argued that the key aspect of this model that enables it to succeed (and that differentiates it from other current models of input coding and lexical matching) is its commitment to position and context-independent representations. Spatial coding solves the problem of how letter position can be conveyed with such codes and also necessitates an approach to lexical matching that is rather different from the dot-product matching approach used in other models. Elsewhere (Davis, 2001, 2004, in press), I have discussed how this model of lexical matching could be implemented, based on the phase coding hypothesis, and the way in which phase coding can represent uncertainty regarding letter position and letter identity.

A number of core principles underlie the spatial coding model, including the principles of abstraction (which commits the model to context-independent representations); invariance (according to which letter position should be coded in a way that preserves information about letter contiguity and the distance between letters); selectivity (according to which word nodes only receive input from relevant letter nodes); translation (which is implemented through the computation of signal-weight differences); harmony (according to which the critical feature of translated letter signals is their congruence; this principle is achieved through the use of superposition matching); clone-equivalence; and one-letter, one-match (the latter two are critical to the model's account of how repeated letters are encoded). Arguments for most of these principles can be made on the basis of general considerations derived from thought experiments. Nevertheless, they can also be subjected to empirical scrutiny. The invariance principle that is implemented here is a slightly modified version of that described by Grossberg (1978; it differs only in that the arithmetic difference between adjacent elements of the spatial code is held to be invariant rather than the ratio of adjacent elements, as in Grossberg's, 1978, version). This principle has been directly challenged by open-bigram coding schemes that dispense with information about the distance between letters (e.g., Grainger & van Heuven, 2003; Grainger & Whitney, 2004; Whitney, 2001). However, as Davis and Bowers (2006) showed, experimental data reinforce the importance of encoding information about letter contiguity. An interesting question for future research concerns the relevance of the above principles for understanding other aspects of cognition that require encoding the serial order of component stimuli, including spoken word identification and short-term memory.

There is now a large body of data on the topic of orthographic input coding. Successfully modeling all of these data is a difficult problem for any model. However, the simulations presented here demonstrate that the spatial coding model does an excellent job of capturing existing data. The model is successful because it addresses each of the three critical processes listed in the introduction: encoding of letter identity and position, lexical matching, and lexical selection. In particular, the model's incorporation of lexical competition as a means of lexical selection elucidates the mechanics of masked priming and helps to illustrate the limitations of conventional methods for studying the processes of encoding and matching, as well as guiding the development of new methodologies to overcome these limitations (Lupker & Davis, 2009).

The spatial coding model shares many features with the SOLAR model of visual word recognition (Davis, 1999). Following a nested modeling approach, future articles will describe other aspects of this model: the model's chunking mechanism, the means by which the model self-organizes, the way in which it learns about word frequency, the way that this biases the identification process, and the way in which the model implements competitive processes to achieve identification of familiar words and learning of new words.

References

Andrews, S. (1989). Frequency and neighborhood effects on lexical access: Activation or search? *Journal of Experimental Psychology: Learning, Memory, and Cognition, 15,* 802–814.

Andrews, S. (1992). Neighborhood effects on lexical access: Lexical similarity or orthographic redundancy? *Journal of Experimental Psychology: Learning, Memory, and Cognition, 18,* 234–254.

Andrews, S. (1996). Lexical retrieval and selection processes: Effects of transposed-letter confusability. *Journal of Memory and Language, 35,* 775–800.

Andrews, S. (2006). All about words: A lexicalist perspective on reading. In S. Andrews (Ed.), *From inkmarks to ideas: Current issues in lexical processing* (pp. 318–348). New York, NY: Psychology Press.

Andrews, S., & Heathcote, A. (2001). Distinguishing common and task-specific processes in word identification: A matter of some moment? *Journal of Experimental Psychology: Learning, Memory, and Cognition, 27,* 514–540.

Averbach, E., & Coriell, A. S. (1961). Short-term memory in vision. *Bell System Technical Journal, 40,* 309–328.

Baayen, R. H., Piepenbrock, R., & van Rijn, H. (1995). The CELEX lexical database (Release 2) [CD-ROM]. Philadelphia, PA: Linguistic Data Consortium, University of Pennsylvania.

Balota, D. A., Cortese, M. J., Sergent-Marshall, S. D., Spieler, D. H., & Yap, M. J. (2004). Visual word recognition of single-syllable words. *Journal of Experimental Psychology, 133,* 283–316.

Balota, D. A., & Spieler, D. H. (1999). Word frequency, repetition, and lexicality effects in word recognition tasks: Beyond measures of central tendency. *Journal of Experimental Psychology-General, 128,* 32–55.

Balota, D. A., & Yap, M. J. (2006). Attentional control and flexible lexical processing: Explorations of the magic moment of word recognition. In S. Andrews (Ed.), *From inkmarks to ideas: Current issues in lexical processing* (pp. 229–258). New York, NY: Psychology Press.

Besner, D., Twilley, L., McCann, R. S., & Seergobin, K. (1990). On the association between connectionism and data: Are a few words necessary? *Psychological Review, 97,* 432–446.

Bowers, J. S. (2002). Challenging the widespread assumption that connectionism and distributed representations go hand-in-hand. *Cognitive Psychology, 45,* 413–445.

Bowers, J. S. (2009). On the biological plausibility of grandmother cells: Implications for neural network theories in psychology and neuroscience. *Psychological Review, 116,* 220–251.

Bowers, J. S. (in press). Does masked and unmasked priming reflect Bayesian inference as implemented in the Bayesian Reader? *European Journal of Cognitive Psychology.*

Bowers, J. S., Damian, M. F., & Davis, C. J. (2009). A fundamental limitation of conjunctive codes in PDP models of cognition: Comments on Botvinick and Plaut (2006). *Psychological Review, 116,* 986–995.

Bowers, J. S., & Davis, C. J. (2004). Is speech perception modular or interactive? *Trends in Cognitive Sciences, 8,* 3–5.

Bowers, J. S., Davis, C. J., & Hanley, D. A. (2005a). Automatic semantic activation of embedded words: Is there a "hat" in "that"? *Journal of Memory and Language, 52,* 131–143.

Bowers, J. S., Davis, C. J., & Hanley, D. A. (2005b). Interfering neighbors: The impact of novel word learning on the identification of visually similar words. *Cognition, 97,* B45–B54.

Bowers, J. S., & Davis, C. J. (2009). Learning representations of word-forms with recurrent networks: Comment on Sibley, Kello, Plaut, & Elman. *Cognitive Science, 33,* 1183–1186.

Bradski, G., Carpenter, G. A., & Grossberg, S. (1994). STORE working memory networks for storage and recall or arbitrary temporal sequences. *Biological Cybernetics, 71,* 469–480.

Bruner, J. S., & O'Dowd, D. (1958). A note on the informativeness of parts of words. *Language and Speech, 1,* 98–101.

Brysbaert, M., & Ghyselinck, M. (2006). The effect of age of acquisition: Partly frequency-related, partly frequency-independent. *Visual Cognition, 13,* 992–1011.

Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing, 37,* 54–115.

Carr, T. H., & Pollatsek, A. (1985). Models of word recognition. In D. Besner, T. G. Waller, & G. E. MacKinnon (Eds.), *Reading research: Advances in theory and practice* (Vol. 5, pp. 2–76). New York, NY: Academic Press.

Chambers, S. M. (1979). Letter and order information in lexical access. *Journal of Verbal Learning and Verbal Behavior, 18,* 225–241.

Christianson, K., Johnson, R. L., & Rayner, K. (2005). Letter transpositions within and across morphemes. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 31,* 1327–1339.

Chung, S. T. L., & Legge, G. E. (2009). Precision of position signals for letters. *Vision Research, 49,* 1948–1960.

Coltheart, M. (2004). Are there lexicons? *Quarterly Journal of Experimental Psychology, 57*A, 1153–1171.

Coltheart, M., Curtis, B., Atkins, P., & Haller, M. (1993). Models of reading aloud: Dual-route and parallel-distributed processing approaches. *Psychological Review, 100,* 589–608.

Coltheart, M., Davelaar, E., Jonasson, J. T., & Besner, D. (1977). Access to the internal lexicon. In S. Dornic (Ed.), *Attention and performance VI* (pp. 535–555). New York, NY: Academic Press.

Coltheart, M., Rastle, K., Perry, C., Langdon, R., & Ziegler, J. (2001). DRC: A dual route cascaded model of visual word recognition and reading aloud. *Psychological Review, 108,* 204–256.

Cortese, M. J., & Fugett, A. (2004). Imageability ratings for 3,000 monosyllabic words. *Behavior Research Methods, Instruments, and Computers, 36,* 384–387.

Cortese, M. J., & Khanna, M. M. (2007). Age of acquisition predicts naming and lexical-decision performance above and beyond 22 other predictor variables: An analysis of 2,342 words. *Quarterly Journal of Experimental Psychology, 60,* 1072–1082.

Cortese, M. J., & Khanna, M. M. (2008). Age of acquisition ratings for 3,000 monosyllabic words. *Behavior Research Methods, 40,* 791–794.

Courrieu, P., Farioli, F., & Grainger, J. (2004). Inverse discrimination time as a perceptual distance for alphabetic characters. *Visual Cognition, 11,* 901–919.

Crepaldi, D., Rastle, K., Davis, C. J., & Lupker, S. J. (2010). *Identification is position-invariant for stem morphemes, but not affixes*. Manuscript in preparation.

Davelaar, E. J. (2007). Sequential retrieval and inhibition of parallel (re)activated representations: A neurocomputational comparison of competitive queuing and resampling models. *Adaptive Behavior, 15,* 51–71.

Davis, C. J. (1999). *The self-organising lexical acquisition and recognition (SOLAR) model of visual word recognition* (Doctoral dissertation). University of New South Wales, Sydney, Australia.

Davis, C. J. (2001, December). *A novel method of parallel matching in neural network models of recognition*. Paper presented at the 10th Australasian Mathematical Psychology Conference, Newcastle, Australia.

Davis, C. J. (2003). Factors underlying masked priming effects in competitive network models of visual word recognition. In S. Kinoshita & S. J. Lupker (Eds.), *Masked priming: The state of the art* (pp. 121–170). Philadelphia, PA: Psychology Press.

Davis, C. J. (2004). Phase coding: A method for simultaneously encoding order and signal strength in a neural network model of recognition. *Proceedings of the Eighth International Conference on Cognitive and Neural Systems,* 21.

Davis, C. J. (2005). N-watch: A program for deriving neighborhood size and other psycholinguistic statistics. *Behavior Research Methods, 37,* 65–70.

Davis, C. J. (2006). Orthographic input coding: A review of behavioural data and current models. In S. Andrews (Ed.), *From inkmarks to ideas: Current issues in lexical processing* (pp. 180–206). Psychology Press.

Davis, C. J. (in press). SOLAR versus SERIOL revisited. *European Journal of Cognitive Psychology.*

Davis, C. J., & Bowers, J. S. (2004). What do letter migration errors reveal about letter position coding in visual word recognition? *Journal of Experimental Psychology: Human Perception & Performance, 30,* 923–941.

Davis, C. J., & Bowers, J. S. (2005, August). *How is letter position coded? Further evidence for a spatial coding model*. Paper presented at the 14th ESCOP conference, Leiden, the Netherlands.

Davis, C. J., & Bowers, J. S. (2006). Contrasting five different theories of letter position coding: Evidence from orthographic similarity effects. *Journal of Experimental Psychology: Human Perception and Performance, 32,* 535–557.

Davis, C. J., & Bowers, J. S. (2010). *Are there neighborhood effects in the lexical decision task?* Manuscript in preparation.

Davis, C. J., Brysbaert, M., Van Der Haegen, & McCormick, S. F. (2009, August). *Computational modelling of visual/anatomical factors affecting letter position coding*. Paper presented at the MathPsych Conference, Amsterdam, the Netherlands.

Davis, C. J., & Lupker, S. J. (2006). Masked inhibitory priming in English: Evidence for lexical inhibition. *Journal of Experimental Psychology: Human Perception & Performance, 32,* 668–687.

Davis, C. J., & Lupker, S. J. (2010). *Masked priming with reversed string primes: Using sandwich priming to contrast two classes of orthographic input coding models*. Manuscript submitted for publication.

Davis, C. J., McCormick, S. F., Van der Haegen, L., & Brysbaert, M. (2010, January). *Factors affecting letter position uncertainty in reading*. Paper presented at the meeting of the Experimental Psychology Society, London, England.

Davis, C. J., Perea, M., & Acha, J. (2009). Re(de)fining the orthographic neighborhood: The role of addition and deletion neighbors in lexical decision and reading. *Journal of Experimental Psychology: Human Perception and Performance.*

Davis, C. J., & Taft, M. (2005). More words in the neighborhood: Interference in lexical decision due to deletion neighbors. *Psychonomic Bulletin & Review, 12,* 904–910.

Dehaene, S., Cohen, L., Sigman, M., & Vinckier, F. (2005). The neural code for written words: A proposal. *Trends in Cognitive Sciences, 9,* 335–341.

Dilkina, K., McClelland, J. L., & Plaut, D. C. (2008). A single-system account of semantic and lexical deficits in five semantic dementia patients. *Cognitive Neuropsychology, 25,* 136–164.

Duñabeitia, J. A., Laka, I., Perea, M., & Carreiras, M. (2009). Is Milkman a superhero like Batman? Constituent morphological priming in compound words. *European Journal of Cognitive Psychology, 21,* 615–640.

Duñabeitia, J. A., Perea, M., & Carreiras, M. (2008). Does darkness lead to happiness? Masked suffix priming effects. *Language and Cognitive Processes, 23,* 1002–1020.

Estes, W. K., Allmeyer, D. H., & Reder, S. M. (1976). Serial position functions for letter identification at brief and extended exposure durations. *Perception and Psychophysics, 19,* 1–15.

Farrell, S., & Lelièvre, A. (2009). End anchoring in short-term order memory. *Journal of Memory and Language, 60,* 209–227.

Fiset, D., Blais, C., Ethier-Majcher, C., Arguin, M., Bub, D., & Gosselin, F. (2008). Features for identification of uppercase and lowercase letters. *Psychological Science, 19,* 1161–1168.

Fodor, J. A. (1975). *The language of thought.* New York, NY: Thomas Y. Crowell

Forster, K. I. (1992). Memory-addressing mechanisms and lexical access. In R. Frost (Ed.), *Orthography, phonology, morphology, and meaning. Advances in psychology,* (Vol. 94, pp. 413–434). Amsterdam, the Netherlands: North-Holland.

Forster, K. I. (1998). The pros and cons of masked priming. *Journal of Psycholinguistic Research, 27,* 203–233.

Forster, K. I., Davis, C., Schoknecht, C., & Carter, R. (1987). Masked priming with graphemically related forms: Repetition or partial activation? *Quarterly Journal of Experimental Psychology, 39,* 211–251.

Forster, K. I., & Shen, D. (1996). No enemies in the neighborhood: Absence of inhibitory neighborhood effects in lexical decision and semantic categorization. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 22,* 696–713.

Frankish, C., & Barnes, L. (2008). Lexical and sublexical processes in the perception of transposed-letter anagrams. *Quarterly Journal of Experimental Psychology, 61,* 381–391.

Frankish, C., & Turner, E. (2007). SIHGT and SUNOD: The role of orthography and phonology in the perception of transposed letter anagrams. *Journal of Memory and Language, 56,* 189–211.

Glezer, L. S., Jiang, X., & Riesenhuber, M. (2009). Evidence for highly selective neuronal tuning to whole words in the "visual word form area." *Neuron, 62,* 199–204.

Gomez, P., Ratcliff, R., & Perea, M. (2008). The overlap model: A model of letter position coding. *Psychological Review, 115,* 577–601.

Grainger, J. (2008). Cracking the orthographic code: An introduction. *Language and Cognitive Processes, 23,* 1–35.

Grainger, J., Granier, J. P., Farioli, F., Van Assche, E., & van Heuven, W. J. B. (2006). Letter position information and printed word perception: The relative-position priming constraint. *Journal of Experimental Psychology: Human Perception and Performance, 32,* 865–884.

Grainger, J., & Jacobs, A. M. (1994). A dual read-out model of word context effects in letter perception: Further investigations of the word superiority effect. *Journal of Experimental Psychology: Human Perception and Performance, 20,* 1158–1176.

Grainger, J., & Jacobs, A. M. (1996). Orthographic processing in visual word recognition: A multiple read-out model. *Psychological Review, 103,* 518–565.

Grainger, J., O'Regan, J. K., Jacobs, A. M., & Segui, J. (1989). On the role of competing word units in visual word recognition: The neighborhood frequency effect. *Perception and Psychophysics, 45,* 189–195.

Grainger, J., & van Heuven, W. (2003). Modeling letter position coding in printed word perception. In P. Bonin (Ed.), *The mental lexicon* (pp. 1–24). New York, NY: Nova Science.

Grainger, J., & Whitney, C. (2004). Does the huamn mnid raed wrods as a wlohe? *Trends in Cognitive Sciences, 8,* 58–59.

Grossberg, S. (1969). Some networks that can learn, remember, and reproduce any number of complicated space-time patterns I. *Journal of Mathematics and Mechanics, 19,* 53–91.

Grossberg, S. (1973). Contour enhancement, short-term memory, and constancies in reverberating neural networks. *Studies in Applied Mathematics, 52,* 213–257.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding: Feedback, expectation, olfaction, and illusions. *Biological Cybernetics, 23,* 187–202.

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* (pp. 233–374). New York, NY: Academic Press.

Guerrera, C., & Forster, K. (2008). Masked form priming with extreme transposition. *Language and Cognitive Processes, 23,* 117–142.

Harm, M. W., & Seidenberg, M. S. (1999). Phonology, reading acquisition, and dyslexia: Insights from connectionist models. *Psychological Review, 106,* 491–528.

Hinton, J., Liversedge, S. P., & Underwood, G. (1998). Neighborhood effects using a partial priming methodology: Guessing or activation? *Journal of Experimental Psychology: Learning, Memory, and Cognition, 24,* 1294–1305.

Holmes, V. M., & Ng, E. (1993). Word-specific knowledge, word-recognition strategies, and spelling ability. *Journal of Memory and Language, 32,* 230–257.

Humphreys, G. W., Evett, L. J., & Quinlan, P. T. (1990). Orthographic Processing in Visual Word Identification. *Cognitive Psychology, 22,* 517–560.

Jacobs, A. M., & Grainger, J. (1992). Testing a semistochastic variant of the interactive activation model in different word recognition experiments. *Journal of Experimental Psychology: Human Perception and Performance, 18,* 1174–1188.

Jacobs, A. M., & Grainger, J. (1994). Models of visual word recognition: Sampling the state of the art. *Journal of Experimental Psychology: Human Perception and Performance, 20,* 1311–1334.

Jacobs, A. M., Rey, A., Ziegler, J. C., & Grainger, J. (1998). MROM-P: An interactive activation, multiple read-out model of orthographic and phonological processes in visual word recognition. In J. Grainger & A. M. Jacobs (Eds.), *Localist connectionist approaches to human cognition* (pp. 147–188). Mahwah, NJ: Erlbaum.

Johnson, R. L. (2007). The flexibility of letter coding: Nonadjacent letter transposition effects in the parafovea. In R. van Gompel, M. Fisher, W. Murray, & R. L. Hill (Eds.), *Eye movements: A window on mind and brain* (pp. 425–440). Oxford, England: Elsevier.

Johnson, R. L., & Dunne, M. D. (2008). *Parafoveal processing of transposed-letter words and nonwords: Evidence against parafoveal lexical activation*. Manuscript submitted for publication.

Johnson, R. L., Perea, M., & Rayner, K. (2007). Transposed-letter effects in reading: Evidence from eye movements and parafoveal preview. *Journal of Experimental Psychology: Human Perception and Performance, 33,* 209–229.

Juhasz, B. J., & Rayner, K. (2003). Investigating the effects of a set of intercorrelated variables on eye fixation durations in reading. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 29,* 1312–1318.

Kinoshita, S., & Norris, D. (2009). Transposed-letter priming of prelexical orthographic representations. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 35,* 1–18.

Kinoshita, S., & Norris, D. (2008). *Letter transposition priming in lexical decision reflects noisy order coding: Or why dyslexia is no substitute for dailysex*. Manuscript submitted for publication.

Lupker, S. J., & Davis, C. J. (2009). Sandwich priming: A method for overcoming the limitations of masked priming by reducing lexical competitor effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 35,* 618–639.

Lupker, S. J., Perea, M., & Davis, C. J. (2008). Transposed-letter effects: Consonants, vowels and letter frequency. *Language and Cognitive Processes, 23,* 93–116.

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

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

McClelland, J. L., & Rumelhart, D. E. (1988). *Explorations in the microstructure of cognition: A handbook of models, programs, and exercises.* Cambridge, MA: MIT Press.

Merikle, P. M., Coltheart, M., & Lowe, D. G. (1971). Selective effects of a patterned masking stimulus. *Canadian Journal of Psychology, 25,* 264–279.

Mewhort, D. J. K., & Campbell, A. J. (1978). Processing spatial information and selective-masking effect. *Perception & Psychophysics, 24,* 93–101.

Monsell, S., (1991). The nature and locus of word frequency effects in reading. In D. Besner & G. W. Humphreys (Eds.), *Basic processes in reading: Visual word recognition* (pp. 148–197). Hillsdale, NJ: Erlbaum.

Murray, W. S., & Forster, K. I. (2004). Serial mechanisms in lexical access: The rank hypothesis. *Psychological Review, 111,* 721–756.

Norris, D. (2006). The Bayesian reader: Explaining word recognition as an optimal Bayesian decision process. *Psychological Review, 113,* 327–357.

Norris, D. (2009). Putting it all together: A unified account of word recognition and reaction-time distributions. *Psychological Review, 116,* 207–216.

Norris, D., & Kinoshita, S. (2008). Perception as evidence accumulation and Bayesian inference: Insights from masked priming. *Journal of Experimental Psychology: General, 137,* 434–455.

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

O'Connor, R. E., & Forster, K. I. (1981). Criterion bias and search sequence bias in word recognition. *Memory & Cognition, 9,* 78–92.

Page, M. P. A. (1994). Modeling the perception of musical sequences with self-organizing neural networks. *Connection Science, 6,* 223–246.

Page, M. P. A. (2000). Connectionist modeling in psychology: A localist manifesto. *Behavioral and Brain Sciences, 23,* 443–467.

Page, M. P. A., & Norris, D. (1998). The primacy model: A new model of immediate serial recall. *Psychological Review, 105,* 761–781.

Pammer, K., Hansen, P., Kringelbach, M., Holliday, I. Barnes, G., Hillebrand, A., . . . & Cornelissen, P. (2004). Visual word recognition: The first half second. *NeuroImage, 22,* 1819–1825.

Pelli, D. G., Burns, C. W., Farell, B., & Moore-Page, D. C. (2006). Feature detection and letter identification. *Vision Research, 46,* 4646–4674.

Perea, M., & Carreiras, M. (2006a). Do transposed-letter similarity effects occur at a prelexical phonological level? *Quarterly Journal of Experimental Psychology, 59,* 1600–1613.

Perea, M., & Carreiras, M. (2006b). Do transposed-letter similarity effects occur at a syllable level? *Experimental Psychology, 53,* 308–315.

Perea, M., & Gotor, A. (1997). Associative and semantic priming effects occur at very short stimulus-onset asynchronies in lexical decision and naming. *Cognition, 62,* 223–240.

Perea, M., & Lupker, S. J. (2003a). Does jugde activate COURT? Transposed-letter confusability effects in masked associative priming. *Memory & Cognition, 31,* 829–841.

Perea, M., & Lupker, S. J. (2003b). Transposed-letter confusability effects in masked form priming. In S. Kinoshita & S. J. Lupker (Eds.), *Masked priming: The state of the art.* Philadelphia, PA: Psychology Press.

Perea, M., & Lupker, S. J. (2004). Can CANISO activate CASINO? Transposed-letter similarity effects with nonadjacent letter positions. *Journal of Memory and Language, 51,* 231–246.

Peressotti, F., & Grainger, J. (1995). Letter position coding in random consonant arrays. *Perception and Psychophysics, 57,* 875–890.

Peressotti, F., & Grainger, J. (1999). The role of letter identity and letter position in orthographic priming. *Perception & Psychophysics, 61,* 691–706.

Perry, C., Ziegler, J. C., & Zorzi, M. (2007). Nested incremental modeling in the development of computational theories: The CDP+ model of reading aloud. *Psychological Review, 114,* 273–315.

Perry, J. R., Lupker, S. J., & Davis, C. J. (2008). An evaluation of the interactive-activation model using masked partial-word priming. *Language and Cognitive Processes, 23,* 36–68.

Plaut, D. C., McClelland, J. L., Seidenberg, M. S., & Patterson, K. (1996). Understanding normal and impaired word reading: Computational principles in quasi-regular domains. *Psychological Review, 103,* 56–115.

Rastle, K. (2007). Visual word recognition. In M. G. Gaskell (Ed.), *The Oxford handbook of psycholinguistics* (pp. 71–87). New York, NY: Oxford University Press.

Ratcliff, R., Gomez, P., & McKoon, G. (2004). A diffusion model account of the lexical decision task. *Psychological Review, 111,* 159–182.

Rayner, K. (1979). Eye guidance in reading: Fixation locations within words. *Perception, 8,* 21–30.

Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin, 124,* 372–422.

Rayner, K., Chace, K. H., Slattery, T. J., & Ashby, J. (2006). Eye movements as reflections of comprehension processes in reading. *Scientific Studies of Reading, 10,* 241–255.

Rayner, K., & Pollatsek, A. (1987). Eye movements in reading: A tutorial review. In M. Coltheart (Ed.), *Attention and performance XII: The psychology of reading* (pp. 327–362). London, England: Erlbaum.

Rayner, K., White, S. J., Johnson, R. L., & Liversedge, S. P. (2006). Raeding wrods with jubmled lettres. *Psychological Science, 17,* 192–193.

Reicher, G. M. (1969). Perceptual recognition as a function of meaningfulness of stimulus material. *Journal of Experimental Psychology, 81,* 275–280.

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

Rubenstein, H., Garfield, L., & Millikan, J. A. (1970). Homographic entries in the internal lexicon. *Journal of Verbal Learning and Verbal Behavior, 9,* 487–494.

Rubenstein, H., Lewis, S. S., & Rubenstein, M. A. (1971). Homographic entries in the internal lexicon: Effects of systematicity and relative frequency of meanings. *Journal of Verbal Learning and Verbal Behavior, 10,* 57–62.

Rumelhart, D. E., & McClelland, J. L. (1982). An interactive activation model of context effects in letter perception: Part II. The contextual enhancement effect and some tests and extensions of the model. *Psychological Review, 89,* 60–94.

Rumelhart, D. E., & Siple, P. (1974). Process of recognizing tachistoscopically presented words. *Psychological Review, 81,* 99–118.

Schoonbaert, S., & Grainger, J. (2004). Letter position coding in printed word perception: Effects of repeated and transposed letters. *Language and Cognitive Processes, 19,* 333–367.

Segui, J., & Grainger, J. (1990). Priming word recognition with orthographic neighbors: Effects of relative prime–target frequency. *Journal of Experimental Psychology: Human Perception and Performance, 16,* 65–76.

Seidenberg, M. S., & McClelland, J. L. (1989). A distributed, developmental model of word recognition and naming. *Psychological Review, 96,* 523–568.

Sereno, S. C., & Rayner, K. (2003). Measuring word recognition in reading: Eye movements and event-related potentials. *Trends in Cognitive Sciences, 7,* 489–493.

Sibley, D. E., Kello, C. T., Plaut, D. C., & Elman, J. L. (2009). Sequence encoders enable large-scale lexical modeling: Reply to Bowers and Davis (2009). *Cognitive Science, 33,* 1187–1191.

Smith, P. T., Jordan, T. R., & Sharma, D. (1991). A connectionist model of visual-word recognition that accounts for interactions between mask size and word-length. *Psychological Research, 53,* 80–87.

Spieler, D. H., & Balota, D. A. (1997). Bringing computational models of word naming down to the item level. *Psychological Science, 8,* 411–416.

Stadthagen-Gonzalez, H., Bowers, J. S., & Damian, M. F. (2004). Age of acquisition effects in visual word recognition: Evidence from expert vocabularies. *Cognition, 93,* B11–B26.

Stadthagen-Gonzalez, H., & Davis, C. J. (2006). The Bristol norms for age of acquisition, imageability and familiarity. *Behavior Research Methods, 38,* 598–605.

Stinchcombe, E. J., Lupker, S. J., & Davis, C. J. (2010). *Transposed letter*

priming effects with masked subset primes: A re-examination of the "relative position priming constraint." Manuscript submitted for publication.

Taft, M. (1991). *Reading and the mental lexicon.* London, England: Erlbaum.

Taft, M., & van Graan, F. (1998). Lack of phonological mediation in a semantic judgment task. *Journal of Memory and Language, 38,* 203–224.

Tydgat, I., & Grainger, J. (2009). Serial position effects in the identification of letters, digits and symbols. *Journal of Experimental Psychology: Human Perception and Performance, 35,* 480–498.

Van Assche, E., & Grainger, J. (2006). A study of relative-position priming with superset primes. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 32,* 399–415.

Van der Haegen, L., Brysbaert, M., & Davis, C. J. (2009). How does interhemispheric communication in visual word recognition work? Deciding between early and late integration accounts of the split fovea theory. *Brain and Language, 108,* 112–121.

van Heuven, W. J. B., Dijkstra, T., Grainger, J., & Schriefers, H. (2001). Shared neighborhood effects in masked orthographic priming. *Psychonomic Bulletin & Review, 8,* 96–101.

Welvaert, M., Farioli, F., & Grainger, J. (2008). Graded effects of number of inserted letters in superset priming. *Experimental Psychology, 55,* 54–63.

Whaley, C. P. (1978). Word–nonword classification time. *Journal of Verbal Learning and Verbal Behavior, 17,* 143–154.

White, S. J., Johnson, R. L., Liversedge, S. P., & Rayner, K. (2008). Eye movements when reading transposed text: The importance of word-beginning letters. *Journal of Experimental Psychology: Human Perception and Performance, 34,* 1261–1276.

Whitney, C. (2001). How the brain encodes the order of letters in a printed word: The SERIOL model and selective literature review. *Psychonomic Bulletin & Review, 8,* 221–243.

Whitney, C. (2004). *Investigations into the neural basis of structured representations* (Doctoral dissertation). University of Maryland, College Park, MD.

Whitney, C., & Grainger, J. (2008). *On coding the position of letters in words: A test of two models*. Manuscript submitted for publication.

Wilson, H. R., & Cowan, J. D. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical Journal, 12,* 1–24.

Zorzi, M., Houghton, G., & Butterworth, B. (1998). Two routes or one in reading aloud? A connectionist dual-process model. *Journal of Experimental Psychology: Human Perception and Performance, 24,* 1131–1161.

Appendix

Simulating the IA Model as a Special Case of the Spatial Coding Model

To specify the original IA model as a special case of the spatial coding model, I replaced Equations 8, 26, and 36a with slightly more complex forms. Thus, Equation 8 is replaced by

$$\text{resPhase}_i(t) = \begin{cases} p^* \text{ such that } S_i(p^*, t) = \max[S_i(p, t)], \\ \quad \text{if PI = true} \\ 0 \quad \text{otherwise.} \end{cases} \quad \text{(A1)}$$

The condition in (A1) is designed to allow position-specific coding models such as the original IA model to be treated as special cases of the spatial coding model. The default setting of the position invariance (PI) switch in the spatial coding model is true. When the PI switch is set to be false, the resonating phase is always zero, which implies that letters signals only contribute to the match to the extent they occur in the expected serial position (where the clause *to the extent* allows for the possibility of some position uncertainty). For example, in Figure 4, the resonating phase is zero in examples A, C, D, E, and F, whether the switch PI is set to be true or false. However, in example B, the resonating phase differs for the position-invariant and position-specific versions. For the position-specific model, PI is false, and combining Equations A1 and 15 implies that $\text{resPhase}_i(t) = 0$ (rather than 3) and $\text{match}_i(t) = 0$ (rather than 1).

Equation 26 is replaced by

$$\text{mismatch}_i(t) = \max\{\text{len}_i, \sum_c \sum_j [\text{act}_{cj}(t)]^{+\text{LWI}}\} - \sum_b [R_{bi}(\text{resPhase}_i, t)]^{\text{LWI}}. \quad \text{(A2)}$$

When LWI = 0, Equation A2 is equivalent to Equation 26. When LWI = 1, Equation A2 implements IA-style letter-word inhibition, subject to the condition that letter activities grow at the same rate in each channel, which is approximately true in the original IA model.

Equation 36a is replaced by

$$\text{wordLetterInput}_{cji}(t) = a_{\text{WL}}[\text{act}_i(t)]^+ / (\sum_j \text{featureLetter}_{cj} + .001)^{\text{TDM}}. \quad \text{(A3)}$$

The parameter TDM acts as a switch. The default value is TDM = 1, so that top-down feedback is modulated by bottom-up input, as in Equation 36a. However, the setting TDM = 0 (so that the strength of top-down feedback is $\alpha_{\text{WL}}[\text{act}_i(t)]^+$, independent of bottom-up input) results in top-down feedback signals that are identical in magnitude to the original IA model.

Thus, to specify the original IA model as a special case of the spatial coding model, the switch PI should be set to false, the letter-word inhibition switch should be set to true (LWI = 1), the top-down feedback modulation switch should be set to false (TDM = 0), end-letter marking should be eliminated by setting $w_{ji}^{\text{initial}} = w_{ji}^{\text{final}} = 0$, and the remaining parameter settings should be as shown in the relevant column of Table 3.

As noted earlier, the IA-style letter-word inhibition implemented in Equation A2 is not exactly equal to the letter-word inhibition of the original IA model because it relies on the assumption that letter activity grows at the same rate in each letter channel. To test the approximation in A2, the identification latencies for the full set of 1,178 words in the original IA vocabulary

(*Appendix continues*)

were compared for the original model and the version of the spatial coding model with the above IA parameter settings. The identification threshold μ was set at .68 (as in all other simulations), and the temporal scaling parameter *dt* was set at .1. The average identification latency was 181.4 cycles for both models. The correlation between the two sets of decision latencies was .999, and the absolute difference in the two latencies for a given word never exceeded one cycle. Thus, the above parameterization is essentially equivalent to the original IA model.

Received September 4, 2009
Revision received January 27, 2010
Accepted January 28, 2010 ■