A.1 Technical Appendix

Here, we present exhaustive technical details of the neural network simulations presented in the paper. The simulations were conducted in the Light Efficient Network Simulator (LENS; for convenience, an archive of the LENS software used for these simulations is available at oddjob.psychology.binghamton.edu) a freely distributed, open source neural network simulator. The details presented here are sufficient to enable replication of the simulations presented in the paper in LENS or any equivalent neural network simulator; additionally, the full simulation code is available from S.L. by request.

Network Architecture

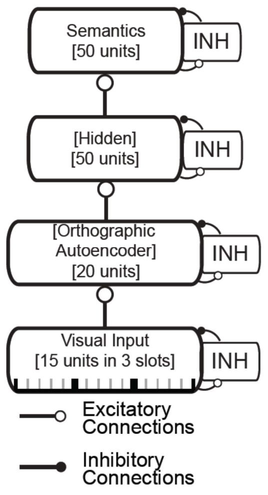

Figure A.1: Network Architecture. This figure is a replication of Figure 1 from the main text.

Description of Each Layer of Units in the Network

Name: Orthographic Inputs
Details: The orthographic input layer consisted of 15 units. Activation of the input units was, in all cases, hard-clamped to either the pattern of activation of a given input (i.e., word, pseudoword, acronym, or illegal string), or the “blank” input (all zeros) presented between items when presenting two items during testing.

Name: hidden[1]
Details: The first hidden layer consisted of 20 units. Activation of units in the hidden[1] layer was determined on a by-unit basis by a multiplicative combination of the standard sigmoidal function (see Equation A.1 below) and the alpha function (see Equation A.2 below).

Name: Orthographic Autoencoder [Output]
Details: The output of the orthographic autoencoder consisted of 15 units. Activation of these units was determined identically as for the units in hidden[1].

Name: hidden[2]
Details: The second hidden layer consisted of 50 units. Activation of units in the hidden[2] layer was determined identically as for the units in hidden[1].

Name: Semantic Output
Details: The semantic output layer consisted of 50 units. Activation of units in the semantic output layer was determined identically as for the units in hidden[1].

Name: Semantic Cleanup
Details: The semantic cleanup layer consisted of 30 units. Activation of units in the semantic cleanup layer was determined identically as for the units in hidden[1].

Name: INH units
Details: The hidden[1], orthographic autoencoder, hidden[2], and semantic output layers each had 1 associated inhibitory (INH) unit. Activation of INH units was determined by the Elbow Function (see Laszlo & Plaut, 2012; Equation A.3 below).

All units, except those in the input and output layers, were input integrating.

Equations

A.1: The Sigmoid Function

$$a = \frac{1}{1+e^{(-i)}}$$

Where:
- a = activation
- i = net input

A.2: The Alpha Function

$$V = \alpha t e^{\frac{-t}{T}}$$

Where:
- V = Voltage scaling factor

α = constant scaling parameter that determines the maximum value of V
t = number of time steps since a unit became active above threshold θ
T = free parameter that determines the time step at which V peaks

A.3: The Elbow Function

$$a = \begin{cases} i & \text{if } i <= h \\ 2i & \text{otherwise} \end{cases}$$

Where
a = activation
i = net input
h = threshold determining the ‘elbow’ point

Activation of inhibitory units is entirely determined by equation A.3. Activation of excitatory units is determined first by applying equation A.1, then multiplying the resultant activation value by a scaling factor determined by equation A.2.

Parameter values used in the model, and their justifications, are given in the “Network Parameters” section below.

Connectivity
The orthographic input layer was fully connected, feedforward, to hidden[1]. These connection weights were constrained to always be > 0.

The hidden[1] layer was fully connected, feedforward, to the orthographic autoencoder output, and to hidden[2]. It was also fully connected, feedforward, to its single associated INH unit. All of these connection weights were constrained to always be > 0.

The $\text{INH}_{\text{hidden[1]}}$ unit was fully connected, feedforward, to hidden[1], but these connection weights were constrained to always be < 0. INH unit connectivity for all following layers of representation is identical to the INH unit connectivity described here.

The orthographic autoencoder output was connected as described above to its single INH unit.

The hidden[2] layer was fully connected, feedforward, to the semantic output layer; these connection weights were constrained to always be > 0. It was also connected as described above to its single INH unit.

The semantic output layer was fully connected, recurrently, to the semantic cleanup layer (not pictured); these connection weights were constrained to always be > 0. It was also connected as described above to its single INH unit.

Network Parameters
Momentum: 0
Learning Rate: Dynamically adjusted during training; initial rate = .001
Number of activation updates (time steps): 50
Elbow function inflection point (h) = .15
$\theta_{autoencoder} = .85$
$\alpha_{autoencoder} = .65$
$T_{autoencoder} = 4$
$\theta_{remainder} = .18$
$\alpha_{remainder} = .33$
$T_{remainder} = 8$

Note, θ, α and T were set separately for the autoencoder and the remainder of the model. In the case of θ, the autoencoder's task is simply easier than the semantic portion of the network's task (i.e., it the autoencoder must learn only systematic relationships, while the semantic network must learn arbitrary relationships). This means that activations tend to be stronger in the autoencoder than in the remainder of the network, which makes a higher θ appropriate. In the case of α and T, this is because the autoencoder was required to complete its task of re-coding orthography faster than the semantic portion of the network was required to complete its task of mapping orthography onto semantics (as is generally assumed to be the case in the ERP literature pertaining to visual word recognition; see Grainger & Holcomb, 2009, for review). In order to ensure that V remains in [0,1] (see Equation A.2), when α changes, T must change as well.

Note that the particular parameter values used here were largely arbitrary and were primarily only subject to the qualitative constraints outlined above. Indeed, exploratory simulations using slightly different parameter values yielded qualitatively similar results to those reported here. Consequently, the performance reported in the present paper is not restricted to only these specific values.

Training
Training was accomplished via back-propagation through time (Rumelhart, Hinton, & Williams, 1986). Error was computed over output layers via cross-entropy (see Hinton, 1989). The use of back-propagation here is not meant as suggesting that back-propagation is a neurally plausible method of reducing error in neural networks. Rather, back-prop is used as a convenience. A method such as contrastive Hebbian Learning (CHL; Ackley, Hinton, & Sejnowski, 1985) is likely to be closer to that used by neurons, however, in some well-defined cases back-prop has been shown to be mathematically identical to CHL (Xie & Seung, 2003), thus producing identical results, and tend to produce similar results even when these exact cases are not met. Consequently, we employed

backpropagation here for convenience, and to leverage the optimizations to the backpropgation algorithm that have been instantiated in our simulation.

The back-propagation algorithm was modified such that excitatory-only connections could not take on weights < 0, and, conversely, such that inhibitory-only connections could not take on weights > 0. This was accomplished by simply adding a condition that stated that, if an illegal weight change was to take place, that weight instead be set to a boundary value (-.0001 or .0001).

The standard back-propagation algorithm (Rumelhart et al., 1986) assumes activation is computed via the sigmoid function. Here, activation was instead computed 1) as a product of the sigmoid function and the alpha function or 2) via the elbow function. Modifications to the back-propagation algorithm were made to account for these changes per the application of the chain rule to the partial derivatives of error in our new activation functions,exactly as was done for the original backpropagation algorithm. Interested readers are referred to the simulation code for the full implementational details.

Error was reduced with no momentum via gradient descent over the error space. The auto-encoder was pre-trained for 20,000 epochs before the remainder of the network was trained; its weights were then frozen in place and the remainder of the network was trained for an additional 15,000 epochs. Inhibitory weights were not trained, but initialized and frozen at small random values; this prevents potential instabilities in the network that can be caused by escalating, counter-acting weight changes in the excitatory and inhibitory units.

During semantic training, examples were clamped on in the input layer, and activation was allowed to propagate for 16 time steps. Error was accumulated over only the final 4 time steps. Note that the model was NEVER trained on repetitions, on nonwords, or in the frequency domain.

Examples
The network was trained ONLY on single presentations of words and acronyms in the temporal domain. The network's vocabulary consisted of three-letter words (65 items) and acronyms (12 items); each of letter of these was represented over 5 units in the input layer (5 units per letter x 3 letters = 15 units). The orthographic structure of the words was CVC, while the orthographic structure of the acronyms required a C in the center position. This was done in order that the orthographic structure of the words and acronyms presented to the model be similar to the orthographic structure of the words and acronyms presented in the ERP experiment. In particular, these structural constraints ensure that the orthographic neighborhood of the acronyms be lower than that of the words; this particular lexical variable is critical in the determination of N400 morphology (see Laszlo & Federmeier, 2011).

Semantic representations associated with visual inputs in the model are 1) sparse and 2) arbitrary. Sparseness is implemented in that no more than 7 out of the 50 semantic units are ever associated with a given input. Arbitrariness is implemented in that, for each visual input, the associated semantic features are chosen randomly. This arbitrariness is justified in that, at least in English, for the type of morphemically simple items used here, orthography-semantics mappings are largely arbitrary (see Plaut, 1997).

During testing, the network was additionally exposed to all the possible nonwords in its vocabulary. These nonwords can further be divided into pseudowords (85 items) and illegal strings (279 items). Like words presented to the model, the orthographic structure of pseudowords is CVC. Like acronyms presented to the model, the orthographic structure of the illegal strings requires a C in the center slot. The reasons for this method of defining pseudowords and illegal strings is identical to that given for the same distinction between words and acronyms above.

Testing
Prior to testing, the connection weights in the network were frozen to their values at the end of training.

During testing, the network was presented with input pairs of the form AA (repetitions) or AB (non-repetitions). Each item of the pair was presented for 16 time steps, with a single time step of blank input between each member of the pair. This resulted in a total presentation period of 33 time steps for each pair. The input was then removed and the network was allowed to return to an inactive resting state over a period of 17 more steps (for a total of 50 time steps for each pair of items). In testing (but *not* in training), the network was *not* re-initialized between items in a pair (but was re-initialized between pairs).

To extract the frequency domain data, we considered several options. The first option was to simply run a FFT analysis on the sERP waveform in an identical manner to the analysis that was applied to the empirical ERP data. However, the model was only run for a total of 50 time steps in the present simulation, leading to fewer samples than in the ERP data, and consequently violating the assumptions of the FFT more severely than is the case for the ERPs.

Fortunately, by virtue of having implemented a computational model that we can probe directly, an alternative and more direct method of generating frequency data was available to us. This is because in the model, we are able to record the activation of individual units directly, instead of only having access waveforms aggregated over numerous neurons as when it is recorded from an electrode on the scalp. That is, in the model, it is possible to *directly* record the activation of each unit and calculate the proportion of these units that fire at different rates. For this reason, we we employed this direct method for probing the response of the model in the frequency domain.

Because model activation/frequency data are on an arbitrary scale, we link the model data to the behavioral data via qualitative inference between the two graphs --- for instance, that the lowest frequencies in the model should be linked to the lowest frequencies in the empirical data.   Different bin sizes for each “level” of frequency were also employed in additional simulations to ensure that the effects that we report are robust to this procedure --- hence, although we would not make a strong claim about a specific value of model frequency corresponding to a specific value of empirical frequency, we have established that lower frequency bins, regardless of how they are defined exactly, always show the effects observed empirically.