A Neurocomputational Model of the N400 and the P600 in Language Processing

Supporting Information

S1 Simulation materials

Table A1 lists the materials used in the simulations.

S2 Derivation of word meaning representations

As word meaning representations, our model employs 100-dimensional binary representations, which were derived from a large corpus of Dutch newspaper texts (the TwNC corpus; Ordelman et al., 2007) using the Correlated Occurrence Analogue to Lexical Semantics (COALS; Rohde et al., 2009).

We first derived a co-occurrence matrix using a 4-word ramped window, meaning that a word a co-occurs with b if a occurs within 4 words to the left or right of b, and that this co-occurrence is weighted by the proximity of a to b on a scale of 4 (direct neighbor) to 1 (separated by three words). This co-occurrence matrix, which we will refer to as X, is constructed for the 15.000 most frequent words. We then pruned all but the 14.000 columns of this matrix, so that the rows of the matrix then represented 14K-dimensional word feature vectors. Next, the weighted frequency of each co-occurrence $w_{a,b}$ of words a and b was normalized by converting it to a pairwise correlation:

$$w'_{a,b} = \frac{T \cdot w_{a,b} - \sum_j w_{a,j} \cdot \sum_i w_{i,b}}{\left(\sum_j w_{a,j} \cdot (T - \sum_j w_{a,j}) \cdot \sum_i w_{i,b} \cdot (T - \sum_i w_{i,b})\right)^{\frac{1}{2}}} \tag{1}$$

where i is a row index, j is a column index, and:

$$T = \sum_i \sum_j w_{i,j} \tag{2}$$

In the resulting matrix, we replaced each negative correlation with 0, and each positive correlation with its square root:

$$norm(w'_{a,b}) = \begin{cases} 0 & \text{if } w'_{a,b} < 0 \\ \sqrt{w'_{a,b}} & \text{otherwise} \end{cases} \tag{3}$$

To obtain the 100-dimensional feature vectors that we used in our simulations, we reduced the dimensionality of the normalized feature vectors by computing the Singular Value Decomposition of the co-occurrence matrix $X_{15000\times14000}$. Here we considered only the first 100 singular values and vectors, such that we obtain matrix $\hat{X}$ that is the best rank-100 approximation to X in terms of sum squared error:

$$\hat{X}_{15000\times14000} = \hat{U}_{15000\times100}\hat{S}_{100\times100}\hat{V}^{T}_{100\times14000} \tag{4}$$

A 100-unit feature vector V_c for a word c is then defined as:

$$V_c = X_c\hat{V}\hat{S}^{-1} \tag{5}$$

which can be converted to a binary vector by setting its negative components to 0, and its positive components to 1.

S3 Details of the training procedure

We trained each model (i.e., one for each simulation) using a two-stage training procedure (see sections 3.2 and 3.3). In both stages, the two models were trained using bounded gradient descent (Rohde, 2002), a modification of the standard backpropagation algorithm (Rumelhart et al., 1986). For each input-target pair c, we minimized the sum squared error E_c between the desired activity d_j and the observed activity y_j for each unit j in the INTEGRATION_OUTPUT layer:

$$E_c = \frac{1}{2}\sum_j (y_j - d_j)^2 \tag{6}$$

Error was reduced by adjusting each weight w_{ij} in the model on the basis of a delta that is proportional to the gradient of that weight, and depends on its previous delta:

$$\Delta w_{ij}(t) = -\varepsilon\rho\frac{\partial E}{\partial w_{ij}} + \alpha\Delta w_{ij}(t-1) \tag{7}$$

where ε is the network's *learning rate*, ρ a *scaling factor* that depends on the length of the entire gradient:

$$\rho = \begin{cases} \frac{1}{||\partial E/\partial w||} & \text{if } ||\partial E/\partial w|| > 1 \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

and α a momentum coefficient, controlling the fraction of the previous weight delta to be added.

The gradient $\frac{\partial E}{\partial w_{ij}}$ of a weight w_{ij}, in turn, is estimated as the product of the *error signal* δ_j of a unit j, and the activation value y_i of a unit i that signals to unit j:

$$\frac{\partial E}{\partial w_{ij}} = \delta_j y_i \tag{9}$$

The error signal δ_j for an output unit j is defined as:

$$\delta_j = (y_j - d_j)(y_j(1-y_j) + 0.1) \tag{10}$$

where the constant 0.1 is a flat spot correction constant (Fahlman, 1988), preventing the derivative $y_j(1-y_j)$ of the sigmoid activation function to approach zero when y_j is near 0 or 1. The error signal δ_j for a hidden unit j, in turn, is defined as:

$$\delta_j = (y_j(1 - y_j) + 0.1) \sum_k \delta_k w_{jk} \tag{11}$$

where all units k are units that receive signals from unit j.

We trained the model for 7000 epochs, in each of which we accumulated gradients over 100 items before updating the weights. Training items were presented in a permuted order, such that by the end of training, the model has seen each item at least 43 times ($7000/(16000/100) = 43.75$). After all of the 16000 items were presented once, the training order was permuted again. Weights were initially randomized within a range of $(-0.25, +0.25)$, and were updated using a learning rate ε of 0.2, which was scaled down to 0.11 with a factor of 0.95 after each 700 epochs (that is, after each 10% interval of the total epochs; $0.2 \times 0.95^{10} \approx 0.11$). The momentum coefficient α was set to a constant of 0.9. Finally, we used a *zero error radius* of 0.1, such that no error was backpropagated if $|y_j - d_j| < 0.1$. The training procedure was identical for stage one and two.

After training, we evaluated the comprehension performance of the model using an output-target similarity matrix. For each item, we computed the cosine similarity between the output vector for that item, and each of the 16000 different target vectors. The cosine similarity between two vectors is defined as:

$$cos(x, y) = \frac{\sum_i x_i \times y_i}{\sqrt(\sum_i x_i^2) \times \sqrt(\sum_i y_i^2)} \tag{12}$$

The output vector for an item was considered correct if it was more similar to its corresponding target vector than to the target vector of any other item. For each of the models and after each training stage, comprehension performance was perfect (100% correct) on the training items. Finally, as the test items are a subset of the training items, comprehension performance was also perfect (100% correct) on the test sets.

S4 Training on perfect word meaning representations

The Retrieval module of our model was trained using a rather non-standard training procedure; we trained it as part of the overall network, rather than as a separate network (see section 3.3.2 for details). We argued that this training procedure is necessary to pressure the model to arrive at a context-sensitive solution in the Retrieval module. Here, we compare the results of this training regime to those obtained with a training procedure in which the Retrieval module is trained on correct word meaning representations (COALS vectors) at the RETRIEVAL_OUTPUT layer (see Table A2). More specifically, we compare the results of our model to four new models, which differ in various architectural aspects. Each of these models is derived by taking the trained Integration module from our model, and then training the Retrieval module on word meaning representations using the same procedure and parameters as discussed above (with the exception that training only lasted 700 epochs, as the models converged faster).

Two of these models have architectures identical to our neurocomputational model (TRUEMODEL), but their Retrieval modules were trained on perfect word meaning representations: the **IntegrationContext** model and the **PerfectIntegrationContext** model. In the **IntegrationContext** model, the contexts in the INTEGRATION_CONTEXT layer depend on the quality of the word meaning representations produced at the RETRIEVAL_OUTPUT layer during training, whereas in the **PerfectIntegrationContext** model these contexts were perfect (i.e., they were recorded from the Integration module). A first thing to note is that both models produce the same P600-effects as our neurocomputational model, which is due the fact that the Integration module is unchanged; only its inputs differ slightly. Neither of them, however, produces the desired pattern of N400-effects; differences between conditions are minimal, and the ordering of N400 estimates is wrong. In a third model, the **RetrievalContext** model, the Retrieval module is trained as a separate SRN with only its own local context (i.e., a RETRIEVAL_CONTEXT layer which receives a copy from the RETRIEVAL layer prior to feedforward propagation, and a RETRIEVAL_CONTEXT → RETRIEVAL projection). Again, whereas this model produces the same P600-effects as our model, it fails to produce the desired N400-effects (minimal differences and incorrect ordering). Finally, the **NoContext** model, is a model in which the RETRIEVAL layer receives no contextual information at all. This model also produces the P600-effects our model produces, but not the N400-effects (again, minimal differences and incorrect ordering).

References

Fahlman, S. E. (1988). An empirical study of learning speed in back-propagation networks. Technical report, Carnegie Mellon University.

Ordelman, R., Jong, F., Hessen, A., and Hondorp, H. (2007). TwNC: A multifaceted dutch news corpus. *ELRA Newsletter*, 12(3-4).

Rohde, D. L. T. (2002). *A connectionist model of sentence comprehension and production*. PhD thesis, Carnegie Mellon University.

Rohde, D. L. T., Gonnerman, L. M., and Plaut, D. C. (2009). An improved model of semantic similarity based on lexical co-occurrence. *Cognitive Science*, pages 1–33.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Table A1: Simulation materials. Overview of the materials used in the simulations. The upper part of this table represents the lexical items used in simulation 1, and the bottom half those in simulation 2.

Sim.	Agent	Patient	NEUTER	Action	Mismatch
1	voetballer *soccer player*	doelpunt *goal*	+	gescoord *scored*	gediend *served*
1	militair *soldier*	land *country*	+	gediend *served*	gescoord *scored*
1	kok *cook*	maaltijd *meal*	-	bereid *prepared*	gezongen *sung*
1	zanger *singer*	lied *song*	+	gezongen *sung*	bereid *prepared*
1	advocaat *lawyer*	bedrijf *company*	+	aangeklaagd *sued*	gelopen *ran*
1	atleet *athlete*	marathon *marathon*	-	gelopen *ran*	aangeklaagd *sued*
1	politicus *politician*	debat *debate*	+	gevoerd *engaged*	uitgegeven *published*
1	uitgever *publisher*	roman *novel*	-	uitgegeven *published*	gevoerd *engaged*
1	arts *doctor*	diagnose *diagnosis*	-	gesteld *made*	geschilderd *painted*
1	schilder *painter*	schilderij *painting*	+	geschilderd *painted*	gesteld *made*
Sim.	**Agent**	**Patient**	NEUTER	**Action**	**Mismatch**
2	rechercheur *detective*	moord *murder case*	-	opgelost *solved*	verhoogd *raised*
2	werkgever *employer*	salaris *salary*	+	verhoogd *raised*	opgelost *solved*
2	dief *thief*	museum *museum*	+	beroofd *robbed*	getrokken *pulled*
2	tandarts *dentist*	tand *tooth*	-	getrokken *pulled*	beroofd *robbed*
2	schipper *sailor*	schip *ship*	+	aangelegd *berthed*	geregisseerd *directed*
2	regisseur *director*	film *movie*	-	geregiseerd *directed*	aangelegd *berthed*
2	piloot *pilot*	vliegtuig *airplane*	+	bestuurd *steered*	afgelegd *taken*
2	student *student*	tentamen *examen*	+	afgelegd *taken*	bestuurd *steered*
2	verzekeraar *insurer*	verzekering *insurance*	-	uitgekeerd *paid*	gereden *rode*
2	wielrenner *cyclist*	etappe *stage*	+	gereden *rode*	uitgekeerd *paid*

Table A2: Comparison of various training regimes for the Retrieval module. Mean N400 and P600 estimates (and standard errors in parentheses) for our neurocomputational model (TrueModel), compared to four different models trained on perfect word meaning representations (COALS vectors). CP = Control (Passive); RA = Reversal (Active); MP = Mismatch (Passive); MA = Mismatch (Active). See text for details on the models.

Model	**Condition**	**Simulation 1**		**Simulation 2**	
		N400	**P600**	**N400**	**P600**
TrueModel	CP	**.438** (.022)	.039 (.006)	**.487** (.010)	.040 (.003)
	RA	**.479** (.011)	.175 (.011)	**.515** (.017)	.145 (.007)
	MP	**.625** (.020)	.228 (.011)	**.609** (.025)	.208 (.020)
	MA	**.564** (.011)	.202 (.009)	**.592** (.021)	.187 (.010)
IntegrationContext	CP	**.355** (.007)	.066 (.007)	**.355** (.006)	.064 (.005)
	RA	**.349** (.008)	.200 (.010)	**.355** (.006)	.165 (.012)
	MP	**.366** (.010)	.230 (.010)	**.352** (.010)	.212 (.020)
	MA	**.368** (.012)	.216 (.009)	**.357** (.010)	.203 (.012)
PerfectIntegrationContext	CP	**.346** (.007)	.066 (.007)	**.351** (.007)	.063 (.005)
	RA	**.340** (.009)	.198 (.010)	**.351** (.007)	.166 (.012)
	MP	**.364** (.009)	.230 (.009)	**.354** (.010)	.214 (.021)
	MA	**.365** (.010)	.216 (.009)	**.362** (.010)	.204 (.012)
RetrievalContext	CP	**.330** (.006)	.064 (.007)	**.356** (.010)	.060 (.005)
	RA	**.333** (.005)	.196 (.010)	**.353** (.010)	.159 (.011)
	MP	**.345** (.007)	.223 (.010)	**.356** (.009)	.207 (.020)
	MA	**.347** (.007)	.208 (.009)	**.353** (.009)	.197 (.012)
NoContext	CP	**.297** (.004)	.064 (.007)	**.315** (.008)	.062 (.005)
	RA	**.297** (.004)	.196 (.010)	**.315** (.008)	.164 (.012)
	MP	**.315** (.007)	.229 (.010)	**.307** (.007)	.211 (.020)
	MA	**.315** (.007)	.213 (.009)	**.307** (.007)	.201 (.011)