UC Merced

Proceedings of the Annual Meeting of the Cognitive Science Society

Title

Strong Systematicity within Connectionism: The Tensor-Recurrent Network

Permalink

https://escholarship.org/uc/item/0rp960zk

Journal

Proceedings of the Annual Meeting of the Cognitive Science Society, 16(0)

Author

Phillips, Steven

Publication Date

1994

Peer reviewed

eScholarship.org Powered by the California Digital Library University of California

Strong Systematicity within Connectionism: The Tensor-Recurrent Network

Steven Phillips
Department of Computer Science
The University of Queensland
Brisbane QLD 4072 Australia
stevep@cs.uq.oz.au

Abstract

Systematicity, the ability to represent and process structurally related objects, is a significant and pervasive property of cognitive behaviour, and clearly evident in language. In the case of Connectionist models that learn from examples, systematicity is generalization over examples sharing a common structure. Although Connectionist models (e.g., the recurrent network and its variants) have demonstrated generalization over structured domains, there has not been a clear demonstration of strong systematicity (i.e., generalization across syntactic position). The tensor has been proposed as a way of representing structured objects, however, there has not been an effective learning mechanism (in the strongly systematic sense) to explain how these representations may be acquired. I address this issue through an analysis of tensor learning dynamics. These ideas are then implemented as the tensor-recurrent network which is shown to exhibit strong systematicity on a simple language task. Finally, it is suggested that the properties of the tensor-recurrent network that give rise to strong systematicity are analogous to the concepts of variables and types in the Classical paradigm.

Introduction

Systematicity is the ability to represent and process structurally related objects such as sentences in the language domain (Fodor & Pylyshyn, 1988). In the case of Connectionist models that learn from examples, systematicity is generalization over objects sharing a common structure. The capacity to generalize over structurally related objects is clearly evident in language where such a vast number of sentences are understood from relatively few examples.

The Classical explanation for systematicity is that cognition is a process of constructing and manipulating symbol structures. Thus, the ability to process a particular symbol structure extends automatically to all sentences conforming to that structure.

Connectionism, by contrast, attempts to explain cognitive behaviour as an emergent property of numeric (non-symbolic) processes. It has been the preoccupation of Connectionists with trying to find the 'right' numeric processes that has lead to the strong criticism of Connectionism as a valid framework for cognitive theories (Fodor & Pylyshyn, 1988). Essentially, the Classical argument is that, without symbol structures and structure sensitive processes one cannot demonstrate systematicity and therefore, one cannot provide an adequate account of cognitive behaviour. See Fodor and Pylyshyn (1988) and Fodor and McLaughlin (1990) for the details of this argument.

Despite this pessimistic conclusion, Connectionists have provided models that generalize over structured domains. For example, Elman's (1990) simple recurrent network and Pollack's (1990) recursive auto-associative memory correctly process sentences not present in the training set. The question is, does this degree of generalization constitute systematicity?

Hadley (1994) concludes *No*. His conclusion was based on a closer examination of six models, which included: McClelland and Kawamoto (1986), Chalmers (1990), Elman (1990), Pollack (1990), Smolensky (1990), and St. John and McClelland (1990). All six models failed to provide clear demonstrations of what Hadley terms, strong systematicity (generalization across syntactic position)[1], for two reasons. Either, the training sets in all probability contained words in all positions; or, representations of words were such that the model presupposes knowledge of syntactic categories, which begs the question of where did this knowledge come from in the first place. See (Hadley, 1994) for more detailed arguments. This second point can also be made with regard to the work of Niklasson (1993).

Although in each case, with the exception of Niklasson, the modelers were not attempting to demonstrate strong systematicity, the result suggests a common limitation of these models. Subsequent, analysis of the information required to correctly position hyperplanes in first-order feedforward and recurrent networks[2] showed that such networks were unlikely to demonstrate strong systematicity when given no *a priori* assumptions about word level similarity (Phillips, 1994). In both cases, there is an independence between the weights that map component objects occurring in one position and the weights that map the same objects occurring in other positions. Consequently, to learn the mapping the networks must, in general, *see* component objects in all positions. This result also applies to the above models, except the tensor, and the models of Brousse and Smolensky (1989) and Phillips and Wiles (1993).

Given the lack of strong systematicity with current network models, it is therefore important to investigate alternative network architectures with a goal of exhibiting strong systematicity. In this paper, the tensor-recurrent network is presented with the motivation of addressing the issue of strong systematicity. The tensor-recurrent network is tested for strong systematicity on a simple language task. The properties of the network that allow strong systematicity are discussed, and finally, some concluding remarks are made regarding the relationship of the network to the Classical paradigm.

[1]For example, correct processing of the sentence *Mary loves John* having only ever seen *John* in the agent position.

[2]Networks without multiplicative weight terms.

The tensor-recurrent network

The simplest task where it is possible to demonstrate generalization across position requires a network to recover upon request the first or second argument of a binary relation (ordered pair). Mathematically, performing this task correctly means implementing a function (f) defined as:

$$\begin{aligned} f(Q_1, (x_i, x_j)) &\rightarrow x_i, \\ f(Q_2, (x_i, x_j)) &\rightarrow x_j, \end{aligned}$$

where Q_1 and Q_2 are question vectors requesting the first and second arguments respectively; and x_i and x_j are elements of a set of atomic objects (i.e., no internal structure nor *a priori* similarity) that together form an ordered pair. If there are N atomic objects then there are N^2 distinct ordered pairs, and therefore the domain of f is a set of $2N^2$ objects. A network is said to exhibit strong systematicity if it can learn to represent the function without having seen every object in both positions. Subsequent analysis of network properties for demonstrating strong systematicity will refer to this function, however, the analysis is easily extended to relations (tuples) of higher order.

Smolensky (1990) showed how a tensor can be used to represent structured objects by the sum of the outer products of vector pairs, the first of which represents a component object, and the second of which represents the role (or relationship) of that component to the structured object. Furthermore, provided the choice of role vectors is orthogonal, each component may be extracted from the tensor representation by performing the inner product of the corresponding role vector with the tensor vector. However, Smolensky's (1987) recirculation algorithm for determining the roles is essentially a three-layer feedforward network similar to the networks of Brousse and Smolensky (1989) and Phillips and Wiles (1993) which Phillips (1994) showed was unlikely to exhibit strong systematicity.

If one assumes the tensor as part of the network's architecture then there are three learning issues that must be addressed. They are, an account of how component representations, role representations and access representations (i.e., the vectors that when applied to the tensor by an inner product return the desired component representation) are acquired through learning. In each case, these three issues are addressed in the tensor-recurrent network by the principle of error backpropagation (Rumelhart, Hinton, & Williams, 1986). That is, by backpropagating an error signal which is a function of the difference between the desired output and the actual output. The novel feature of this network is that by backpropagating the error signal through the *tensor* units component, role and access vector representations are learnt in response to the demands of the task. The explanation for the motivation for this network that follows refers to the graphical description of the network which is given in Figure 1.

Component representations

The tensor scheme assumes some representation of components which are bound (by an outer product) to a representation of their associated roles. The outer product of vectors $\vec{V}$ and $\vec{W}$ is defined as: $T_{ij} = V_i.W_j$, where T_{ij} is the ith-row jth-column element of the resulting rank-2 tensor $\vec{T}$; and V_i and W_j are the ith and jth elements of vectors $\vec{V}$ and $\vec{W}$, respectively. By applying the inner product of the role vector and the tensor representation, the original component representation can be extracted. The inner product of tensor $\vec{T}$ with vector $\vec{W}$ is defined as: $V_i = \sum_j^N T_{ij}.W_j$, where V_i is the ith element of the resulting vector $\vec{V}$; T_{ij} is the ith-row jth-column element of tensor $\vec{T}$; and W_j are the jth element of the N-dimensional vector $\vec{W}$.

A tensor representation scheme assumes that the input and output modalities (i.e., object representations) are the same. However, the input representation of an object, for example *John*, could be a string of letters, whereas the output representation could be a sequence of phonemes.

This issue is addressed in the tensor-recurrent network by requiring the network to auto-associate[3] the input. Auto-association is achieved by mapping the input to a set of *hidden* units; performing the outer product of the resulting vector with the generated *role* vector; adding the result to the current representation held in the *tensor* units; performing the inner product of the resulting tensor representation with the same role vector; and finally, mapping the result represented at the *inner product* units to the *output* units. In the case where the role vectors are mutually orthogonal and of length 1, then by the two mathematical laws: $(\vec{v} \otimes \vec{w}) \odot \vec{w} = \vec{v}$, if $\|\vec{w}\| = 1$, and $(\vec{v} \otimes \vec{w}) \odot \vec{z} = \vec{0}$, if $\vec{w} \perp \vec{z}$, the vector representation at the *hidden* units will be the same as the vector representation at the *inner product* units. Consequently, the network will act like a three-layer feedforward network, which will, in principle, allow any mapping from input to output (i.e., be modality independent) when error is backpropagated from the output units through the *tensor* units to the *input* units. It is therefore possible to address situations where the input and output representations of an object are different, which is the case when subjects are asked to provide verbal descriptions of scenes, for example.

Role representations

The use of a tensor also assumes the existence of role vectors which must be made orthogonal for the components to be subsequently extracted without interference. Rather than rely on external agents (which ultimately must be explained) each *role* vector in this network is generated by an input vector (via some transformations) and the previous *state* vector (which is currently held in the *context* units)[4]. Suppose the network has already been given the first component x_1, and is currently required to auto-associate the second component x_2 (i.e., x_2 is the current target). Then, the error at the output layer is:

$$\begin{aligned} E &= \|\vec{x}_2 - g((f(\vec{x}_1) \otimes \vec{R}_1 + f(\vec{x}_2) \otimes \vec{R}_2) \odot \vec{R}_2)\| \\ E &= \|\vec{x}_2 - g(\alpha f(\vec{x}_1) + f(\vec{x}_2))\|, \end{aligned}$$

where f is the function from *input* to *hidden* units; g is the function from *inner product* to *output* units; $\vec{x}_1$ and $\vec{x}_2$ are the input/output representations of the first and second component objects, respectively; $\vec{R}_1$ and $\vec{R}_2$ are the respective role

[3]In the sense that it is associating the same object, but with potentially different input and output representations.

[4]Essentially, this part of the network is the same as Elman's (1990) simple recurrent network.

vectors; and, α is just the dot product of $\vec{R}_1$ and $\vec{R}_2$, respectively. Thus, error E goes to zero ($E \rightarrow 0$) when $\alpha \rightarrow 0$. That is, when the two role vectors are orthogonal, assuming $g(f(\vec{x}_2)) = \vec{x}_2$ (i.e., the network has learnt to auto-associate the x_2 component).

Access representations

Given that the tensor holds a representation of a structured object there remains the issue of extracting component representations. Typically, the access vector is provided by some external agent as the role vector to which the desired component was associated. However, if the role vectors are generated internally, then there is no reason to expect the access vectors (supplied as input) will be the same. Here, the information requesting one of the components presented at the *question* units is mapped to the *cue* units. The inner product of vectors at the *cue* and *tensor* units results in a vector at the *inner product* units which is then mapped to the output units. Again, using the principle of error backpropagation, the appropriate access vectors (at the *cue* units) are learnt by backpropagating the error at the output layer through the *tensor* units to the *question* layer. Suppose the tensor currently holds a representation of the ordered pair (x_1, x_2) and a request is made for the first component. Then, the error at the output layer is:

$$\begin{aligned} E &= \|\vec{x}_1 - g((f(\vec{x}_1) \otimes \vec{R}_1 + f(\vec{x}_2) \otimes \vec{R}_2) \odot \vec{Q}_1)\| \\ E &= \|\vec{x}_1 - g(\alpha f(\vec{x}_1) + \beta f(\vec{x}_2))\|, \end{aligned}$$

where $\vec{Q}_1$ is the question vector at the *cue* units; and, α and β are the dot products of $\vec{Q}_1$ with $\vec{R}_1$ and $\vec{R}_2$, respectively. Error E goes to zero ($E \rightarrow 0$) when $\alpha \rightarrow 1$, and $\beta \rightarrow 0$ (i.e., when the question vector is collinear with the first role vector, assuming $g(f(\vec{x}_1)) = \vec{x}_1$.

Evaluating the network

The analysis in the previous section assumed various learning principles (e.g., auto-association of inputs and orthogonalization of role vectors). In this section, these principles are tested on a simple language task.

Task

In this task, simple sentences conform to the structure *agent-action-patient*. Each component is presented to the network one per time step after which the network is given one of three possible questions: *Who performed the action?*; *What was the action?*; and *What was affected by the action?*, from which the network should respond with the *agent*, *action*, and *patient*, respectively. For example, the input *John loves Mary. Who is loved?* would be a sequence of four vectors representing *John*, *loves*, *Mary*, *Who is loved?* (respectively), and presented in that order to the network. At the fourth time step the correct response is a vector representing *Mary* at the output layer.

The *action* and *patient* components are drawn from the set {*Bill*, *John*, *Karen*, *Mark*, *Vivian*}, and the *action* component is drawn from the set {*calls*, *chases*, *loves*}. All components are encoded locally (i.e., by orthogonal vectors where one unit has a value of 1 and the rest 0). The network is said to have demonstrated strong systematicity if having only seen, for example, *Mary* in the *agent* position in the training set generalize to cases where *Mary* is in the *patient* position in the test set.

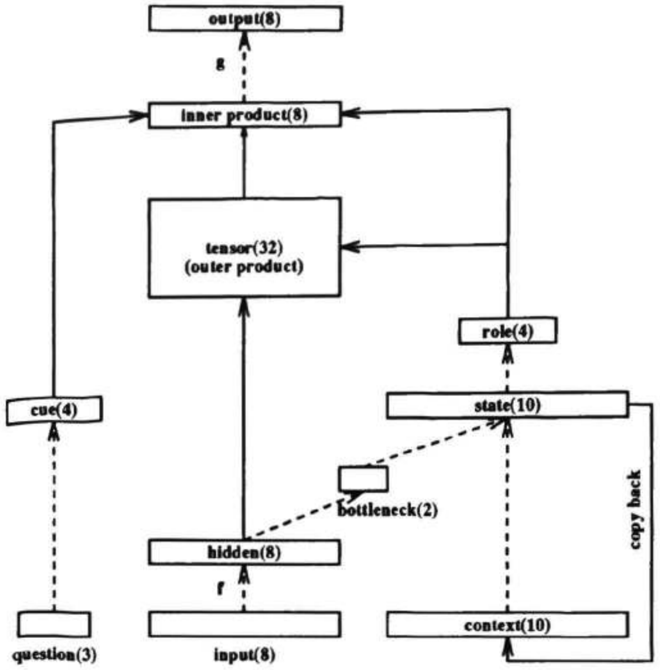

Figure 1: The tensor-recurrent network architecture. Dotted arrows indicate completely connected modifiable weights. In this case, the destination units have *tanh* as their activation function. Solid arrows indicate fixed connections of weight 1. In this case, the activation function of the destination units is the *identity* function. The connectivity of the fixed, weight 1 connections is such as to implement the operator as shown (i.e., inner product, outer product, and copy). Parenthesized values indicate number of units used in simulations.

Method

Each trial, starting from a randomly initialized set of weights, consisted of training on 20 randomly generating sentence/question sequences, and testing on a separate set of 100 randomly generated sequences. Importantly, since generalization across position is being tested, the training and testing distributions are not the same (i.e., some nouns should not appear in both positions in the training set, but should appear in the test set). Each word or question was presented one per time step with the network activations being reset to zero at the beginning of each sentence. The network was trained using the standard backpropagation algorithm (Rumelhart et al., 1986) using the sum of squares error function with a learning rate of 0.1, until the activation of every output units was within 0.4 of the target output for all patterns in all sequences. Two criteria were used on the test sequences. A correct response to the question vector was considered when 1) the maximally activated output unit corresponded to the unit with target 1 (maximum criterion); and 2) all output units were within 0.5 of their target activations. The amount of overlap between the *agent* and *patient* positions was varied from 0 (no noun appeared in both positions) to 5 (every noun appeared in both positions). On each trial the number of correct responses to the question in the test set was recorded. (NB. During the test phase performance on the auto-association of input was not considered. Its purpose was to encourage the formation

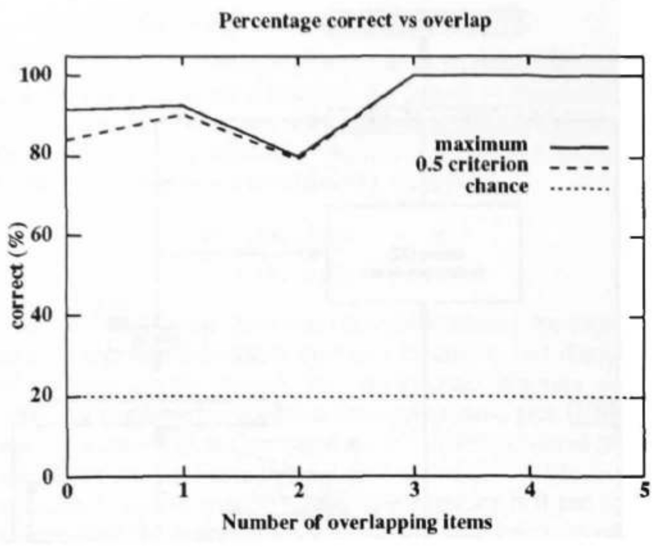

Figure 2: Percentage correct with varying degrees of overlap each averaged over 5 trials.

of internal representations during training, as discussed in the previous section.)

Results

The percentage of correct responses to the question vector on the test set averaged over 5 trials is given in Figure 2, where the bottom dotted line indicates chance level response. The graph shows results only for noun-position combinations that did not occur in the training set. For example, when no noun appeared in both agent and patient positions in the training set, the network achieved 92% accuracy (maximum criterion) and 83% (0.5 criterion) on the test set. When three or more of the possible five nouns appear in both positions in the training set, the network was 100% accurate on the test set.

Discussion

The point of the simulation was to demonstrate a network that could exhibit strong systematicity (i.e., generalization across position). The simulation results showed that it was not necessary for the network to have seen every noun in both positions in the training set (i.e., the tensor-recurrent network is strongly systematic with respect to this task).

Strong systematicity was a consequence of separating the responsibility for representing component and position information. Partly, this separation was due to the architecture whereby component information was represented (independent of its position) at the *hidden* units and position information was represented (independent of its component value) at the *role* units. The component-independence of position information was encouraged by the *bottleneck* units which attenuate the affect of the current input on the state.

However, the separation of component and position information was also a result of the network's learning dynamics. Crucial for correct extraction of components is the orthogonalization of the role vectors. Table 1 shows how with learning the role vectors become progressively more orthogonal. (The measure being one minus the dot product so that zero implies collinear and one implies orthogonal for non-zero vectors.) Even after 1000 epochs of training the *action* and *patient* role vectors were collinear. However, they were sufficiently close[5] to orthogonal by epoch 4900. This table shows that before training the network was not systematic. The collinearity of the *agent* and *patient* role vectors, for example, means that the network could not extract the associated verb without extracting the noun in the *patient* position. Therefore, the strong systematicity exhibited by the network was also a consequence of training.

The performance of the network on training sets where less than three objects appeared in both positions was well above chance level, but it was not perfect. The likely cause of these errors was that the generated role vectors were sensitive to the input value, as well as the position. For example, if the role vector resulting from *John* in the first position was significantly different from the role vector resulting from *Mary* in the first position, then correct extraction would require two different first position cue vectors. But it is not possible for the network to generate two different vectors for the first position cue from the same input question.

The lack of perfect generalization raises the question of just what degree of generalization across position is desirable. For example, how much overlap should there be in the training set before one can expect generalization across position, and what level of generalization should we then expect? In the case of the tensor-recurrent network, a high degree of generalization occurred when there was no overlap between agent and patient positions. The network also generalized to nouns that appeared in the verb position, which suggests that the network was too systematic.

These questions, which fall into the domain of natural language acquisition, are not addressed by Hadley's strong systematicity definition. The importance of Hadley's work has been to identify and specify a qualitative degree of generalization that is evident in people, but not in previous Connectionist models, and thereby suggesting that there is some common property lacking in these models.

Phillips (1994) argued that these models lacked a dependency between the weights that implement the mapping of objects in their various positions. This dependency property was implemented in the tensor-recurrent network by using the same set of weights that map between objects and internal representations in all positions, and by presenting the component objects temporally. Presenting all components spatially, such as in the feedforward networks of Brousse and Smolensky (1989) and Phillips and Wiles (1993), requires a different set of units and weights. The independency between these weights means the network must see all objects in all positions.

Concluding remarks

The properties sufficient to demonstrate strong systematicity with respect to this sentence-question task were: 1) multiple copies of internal representation subspaces provided by the tensor so that learning to represent a component in one subspace automatically transfers to all other subspaces, since

[5]It is not necessary that the role vectors be perfectly orthogonal since the non-linear activation function at the output layer masks out residual vectors. Similarly, the access vectors at the *cue* units were not the same as the role vectors.

these subspaces are the domain and co-domains of functions realized by the same set of weights; and 2) the facility, provided by the inner product operator, to select a subspace independent of the representation it currently contains. In the Classical paradigm, the second property is analogous to a *variable* where vectors in the tensor-recurrent network's role space are like variable identifiers, and the first property is analogous to a *type* (i.e., the set of allowable instances).

Traditionally, variable identifiers have been discrete objects serving only to distinguish one variable from another. Discrete objects are, in general, not learnable by a hill-climbing strategy which requires a continuously differentiable surface. What is interesting about the tensor-recurrent network solution is that by defining variable identifiers (role vectors) over a continuous space the identifiers were learnable by a hill-climbing stategy which is characteristic of the Connectionist approach to the acquisition of behaviour. It suggests how symbolic structures may arise, in part (as the inner and outer product operators were built-in), out of structure insensitive processes such as gradient-descent. However, the task used here was very simple. The extent to which this is possible will depend on exhibiting properties like strong systematicity in more complex domains.

Table 1: Orthogonality between agent, action and patient role vectors measured as one minus the dot product.

Update	agt-act	agt-pat	act-pat	aver.
0	0.32	0.22	0.02	0.19
1000	0.89	0.96	0.00	0.61
4900	0.70	0.59	0.71	0.67

Acknowledgments

I would like to thank Janet Wiles and Simon Dennis for helpful discussions relating to this work, and Janet Wiles, Kathleen Delling-Eiselt and two anonymous reviewers for their comments on previous drafts of this paper. The author was supported by a University of Queensland Postgraduate Research Award.

References

Brousse, O. J., & Smolensky, P. (1989). Virtual memories and massive generalization in connectionist combinatorial learning. In *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, pp. 380–387. NJ: Lawrence Erlbaum.

Chalmers, D. J. (1990). Syntactic transformations on distributed representations. *Connection Science*, *2*, 53–62.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Fodor, J. A., & McLaughlin, B. P. (1990). Connectionism and the problem of systematicity: Why Smolensky's solution doesn't work. *Cognition*, *35*, 183–204.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, *28*, 3–71.

Hadley, R. F. (1994). Systematicity in connectionist language learning. *Mind and Language*. To appear.

McClelland, J. L., & Kawamoto, A. H. (1986). *Mechanisms of Sentence Processing: Assigning roles to Constituents of Sentences*, Vol. 2 of *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, chap. 19, pp. 272–331. MIT Press, MA.

Niklasson, L. F. (1993). Structure sensitivity in connectionist models. In *Pre-proceedings of the Connectionist Summer School*.

Phillips, S. (1994). Systematicity and connectionism. In Tsoi, A. C., & Downs, T. (Eds.), *Proceedings of the Fifth Australian Conference on Neural Networks*, pp. 53–55. University of Queensland Electrical and Computer Engineering.

Phillips, S., & Wiles, J. (1993). Exponential generalizations from a polynomial number of examples in a combinatorial domain. In *Proceedings of the International Joint Conference on Neural Networks*, pp. 505–508 Nagoya, Japan.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, *46*, 77–105.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning Internal Representations by Error Propagation*, Vol. 1 of *Parallel Distributed Processing*, chap. 8, pp. 319–362. MIT Press.

Smolensky, P. (1987). On variable binding and the representation of symbolic structures in connectionist systems. Technical Report CU-CS-355-87, University of Colorado at Boulder.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, *46*, 159–216.

St. John, M. F., & McClelland, J. L. (1990). Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence*, *46*, 217–257.