Natural Language Processing with Subsymbolic Neural Networks *

Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
risto@cs.utexas.edu

1 Introduction

Natural language processing appears on the surface to be a strongly symbolic activity. Words are symbols that stand for objects and concepts in the real world, and they are put together into sentences that obey well-specified grammar rules. It is no surprise that for several decades natural language processing research has been dominated by the symbolic approach. Linguists have focused on describing language systems based on versions of the Universal Grammar. Artificial Intelligence researchers have built large programs where linguistic and world knowledge is expressed in symbolic structures, usually in LISP. Relatively little attention has been paid to various cognitive effects in language processing. Human language users perform differently from their linguistic competence, that is, from their knowledge of how to communicate correctly using language. Some linguistic structures (such as deep embeddings) are harder to deal with than others. People make mistakes when they speak, but fortunately it is not that hard to understand language that is ungrammatical or cluttered with errors. Linguistic and symbolic artificial intelligence theories have little to say about where such effects come from. Yet if one wants to build machines that would communicate naturally with people, it is important to understand and model cognitive effects in natural language processing.

The subsymbolic neural network approach holds a lot of promise for modeling the cognitive foundations of language processing. Instead of symbols, the approach is based on distributed representations that represent statistical regularities in language. Many cognitive effects arise naturally from such representations. In this chapter, the subsymbolic approach is first contrasted with the symbolic approach. Properties of distributed representations are illustrated, and examples of cognitive effects that arise are given. The achievements of the approach, in terms of subsymbolic systems that have been built so far, are reviewed and some remaining research issues outlined.

*To appear in A. Browne (editor), *Neural Network Perspectives on Cognition and Adaptive Robotics*. Bristol, UK; Philadelphia, PA: Institute of Physics Press, 1997

```
(car-32
  (owner (person-22
           (name John)
           (age 42)
           (married T)))
  (color red)
  (type sports)
  (age 5))
```

Figure 1: **Symbolic representation of a concept.** The representation consists of discrete and disjoint symbols, organized in a list structure that is grammatical and concatenative.

2 Subsymbolic Representations

Symbolic and subsymbolic natural language processing systems are based on different strategies for representing information. In this section, the symbolic strategy is first contrasted with the subsymbolic (also called distributed) approach. Properties of distributed representations are then illustrated in a basic sentence case-role assignment task.

2.1 Properties of Subsymbolic Representations

Figure 1 shows a typical symbolic representation for an object, in this case car-32, expressed in LISP syntax. Such a representation could reside in the memory of a symbolic natural language processin (NLP program, as part of its world knowledge. Several observations can be made on this representation. First, the symbols are discrete and disjoint. The symbol is either there or not, and if it is there, it is the symbol exactly. In other words, it is not possible for a symbol to exist at e.g. 50% strength, or 90% accuracy. Second, the symbolic structure is grammatical. There are rules on how to read it: the first element in the list is the name of the whole list, and the following elements are slot-filler pairs that describe the object. Third, the structure is compositional and concatenative. It is possible to change the representation by just adding, changing, or deleting parts of it, and such changes do not affect any other part of the representation. For example, the owner might have been specified earlier as just `John`, but that symbol was replaced by a whole structure describing John. Such a change had no effect on any other part of the representation of car-32, or representations of any other objects in memory.

Figure 2 shows how such concepts could be represented subsymbolically in a neural network. Each concept is a *different pattern of activity* over the same set of units, and the processing knowledge about these concepts is superimposed on the same set of connection weights. A single unit does not represent any particular item, nor does an individual connection weight stand for any particular relation. Each unit and connection is involved in representing many different pieces of information. They stand for entities of finer granularity than symbols; therefore, representation and processing is subsymbolic.[1]

Subsymbolic (i.e. distributed) representations have properties that are very different from the symbolic representations: (1) The representations are continuously valued; (2) Similar concepts have similar representations; (3) The representations are holographic, that is, any part of the

[1]Subsymbolic representations are also often called distributed representations, and each unit a microfeature. This is to contrast them with neural network implementations of symbolic representations, where each unit represents a separate item, or a separate semantic feature.

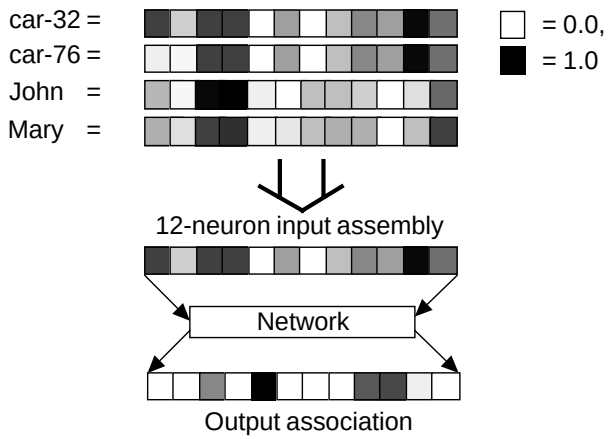

Figure 2: **Subsymbolic representation.** Each item is represented as a pattern of activity over the same set of units, and the information about how to process the patterns is superimposed on the same set of connection weights. The patterns are continuous and can be more or less accurate. Similar concepts are represented by similar patterns, and the pattern as a whole is meaningful, not the values of individual units. The values are indicated by gray-scale coding in the figure. The output association could represent, for example, properties associated with the concept, such as name, age, type, color (cf. figure 1).

representation can be used to reconstruct the whole; and (4) Several different pieces of knowledge are superimposed on the same finite hardware.

From the first two properties it follows that the representations can reflect the meanings of the concepts for which they stand. Similar meanings have similar representations. Because they are continuous, it is possible to represent different degrees of similarity, and category memberships become a matter of degree. There are no clear-cut symbols, because representations belong to each and every class to a different degree, depending on their similarities to other representations in the class.

The holographic property (3) makes the system robust against noise, damage, and incomplete information. Because the same information is represented in several places, the processing is effectively based on an average of several representations. Noise is automatically filtered out in the averaging process, and loss of a few processing elements does not affect the average very much. The system does not selectively lose discrete blocks of information; instead, the accuracy of the output gradually degrades. Even when the input pattern is incomplete, the system can use the rest of the pattern to reconstruct the missing information. This property can be used in high-level systems to automatically create defaults and expectations.

The fourth property results in spontaneous generalization. This is the most important and most striking difference from the symbolic approach. It is not possible to change the representation encoded in a subsymbolic network without affecting the representations of all other items. Knowledge about an item is automatically generalized to all other items, to the degree that they are similar to that item. For example, when a new fact is learned about car-32, it is coded into the network by changing the connection weights. When the pattern for car-32 is input, the output of the network now shows the new fact. Because the weight changes are small and distributed over the entire set of weights, the output for `John` remains largely unchanged. This is because the input pattern for car-32 is very different from the pattern for John. The individual changes made in encoding the new fact about car-32 have mostly a random effect on `John`, canceling out on the average. However, the pattern for car-76 is very similar to `car-32`. When `car-76` is input to the network, the weight

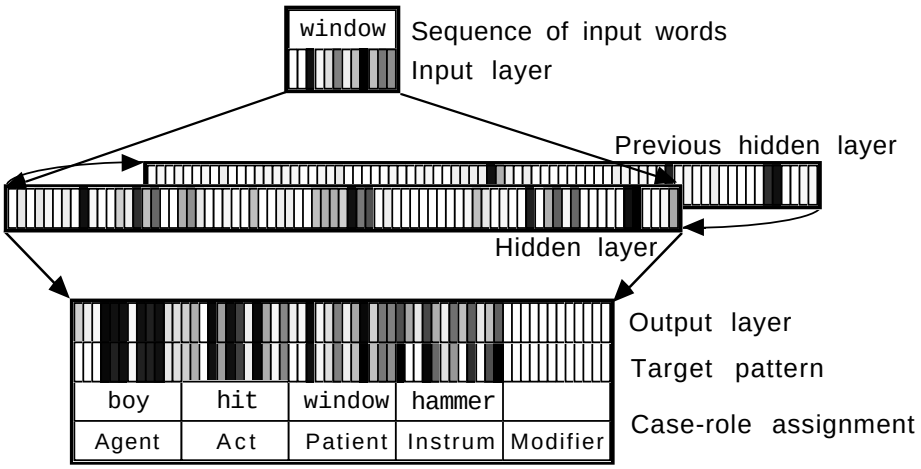

Figure 3: **Subsymbolic case-role assignment of simple sentences.** The network is a Simple Recurrent Network that modifies word representations at its input layer as part of the learning process. In this snapshot, the network is in the middle of reading `The boy hit the window with the hammer.`

changes correlate with the input pattern very well, and the output for `car-76` now also shows the new fact. In other words, the new fact about car-32 is automatically generalized to car-76, to the degree that the representation for car-76 is similar to that for car-32.

2.2 Example: Subsymbolic Representations in Sentence Case-Role Assignment

Reading an input sentence into an internal representation is a most fundamental task in natural language processing. Below, properties of the subsymbolic representations will be illustrated in this task. The internal representation is based on the theory of thematic case roles (Fillmore 1968; Cook 1989). The task consists of reading in the sentence word by word, and deciding which words play the roles of the act, agent, patient, instrument, and patient-modifier in the sentence.

For example, in `The ball hit the girl with the dog`, the subject `ball` is the instrument of the `hit` act, the object `girl` is the patient, the with-clause `dog` is a modifier of the patient, and the agent of the act is unknown. Role assignment is context-dependent: in `The ball moved`, the same subject `ball` is the patient. Assignment also depends on the semantic properties of the word. In `The man ate the pasta with cheese`, the with-clause modifies the patient; but in `The man ate the pasta with a fork`, the with-clause is the instrument. In yet other cases, the assignment must remain ambiguous. In `The boy hit the girl with the ball`, there is no way of telling whether `ball` is an instrument of `hit` or a modifier of `girl`.

The architecture of the case-role assignment network is shown in figure 3. This network has one input assembly, for the current word in the input sequence, and five assemblies in the output, one for each case role. Through the backpropagation learning algorithm it is trained to map a sequence of input word representations into a stationary case-role representation. The network structure is based on Elman's (1990, 1991a) Simple Recurrent Network (SRN). A copy of the hidden layer at time step t is saved and used along with the actual input at step $t+1$ as input to the hidden layer. The previous hidden layer serves as a sequence memory, keeping track of where in the sequence the parser currently is and what has occurred before. The error is backpropagated and weights are changed at each step.

At the same time as the network learns the case-role assignment task, it develops distributed representations for the words. This is done through method called FGREP (Miikkulainen 1993; Miikkulainen and Dyer 1991). Initially the representations for all words are assigned to random patterns. The representations are stored in a lexicon outside the network, and input and target patterns are formed by concatenating word representations currently in the lexicon. During each training presentation, the backpropagation error signal is propagated to the input layer, and the representations are changed as if they were an extra layer of weights. In other words, the network tries to modify the representations so that it could better perform the case-role assignment task. As a result, the final representations effectively code properties of the input words that are most crucial to the task.

The FGREP method allows the network to develop its own distributed input/output representations, a task which would be difficult to do by hand. However, it is important to note that backpropagation neural networks in general always develop distributed representations in their hidden layers as part of the learning. Such representations are learned essentially in the same process as the FGREP representations, and therefore the analysis below illustrates the properties of internal network representations in general.

The network was trained with data designed by McClelland and Kawamoto (1986) for the case-role assignment task. There were 19 sentence templates of the type `the human ate the food`. The actual sentences were formed by replacing the category words `human` and `food` with the actual members of the category, such as `man`, `woman`, `boy`, `girl`, and `chicken`, `cheese`, `pasta`, `carrot`. There were a total of 1,475 sentences, with 30 different words. The network of course did know about the templates and the categories, it only saw the actual sentences during training. To do the case-role assignment correctly, it had to learn the regularities in the examples and code them into the network weights and word representations.

The network successfully learned the case-role assignment task, and the representations converged to a set where words that are used the same way in the data have similar representations (see example set in figure 4). One way to visualize the similarities among the 12-dimensional vectors is through a hierarchical merge clustering algorithm (figure 5). This process starts from the set of representation vectors. At each step, two representations closest to each other are found and merged into a single representation. In the process, clusters of similar representations become visible. The clusters found this way turn out quite similar to the noun categories that were used in generating the sentences. There are six prominent clusters with very small distances between items: animals, humans, utensils, two different types of hitters, and a combination of foods and fragile-objects. Ambiguous words (`chicken`, which is an animal and food, and `bat`, an animal and hitter) and words with an unusual use (e.g. `dog`, which is a possession but not a hitter) do not fit very well into any category, and they are merged later in the process. The distances between the six clusters are quite large, indicating that they are well separated and spread out in the representation space. The cluster analysis demonstrates that the system develops subsymbolic representations that stand for the meanings of the words. Words that belong to similar categories have similar representations.

Inspection of the representations in figure 4 suggests that a single component does not play a crucial role in the classification of items. The fact that a word belongs to a certain category is indicated by the activity profile as a whole, instead of particular units being on or off. In other words, information is distributed over the entire representation vector in a holographic fashion. This is verified by inspection. The whole categorization is visible even in the values of a single component (figure 6). Each component provides a slightly different perspective on the data, and combining the values of more components provides an even more descriptive categorization. The complete

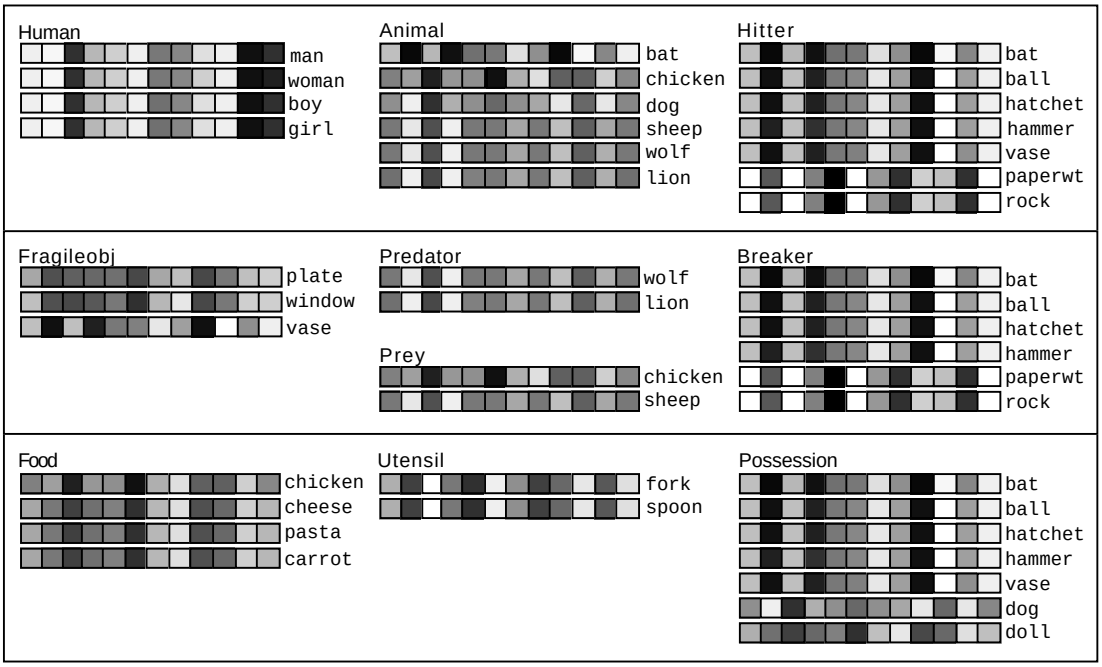

Figure 4: **Distributed representations developed in the case-role assignment task.** The representations are grouped according to categories used in generating the sentences. The representations were initially random. The network learned that certain words are used in similar ways in the data, and coded the similarities in the representations.

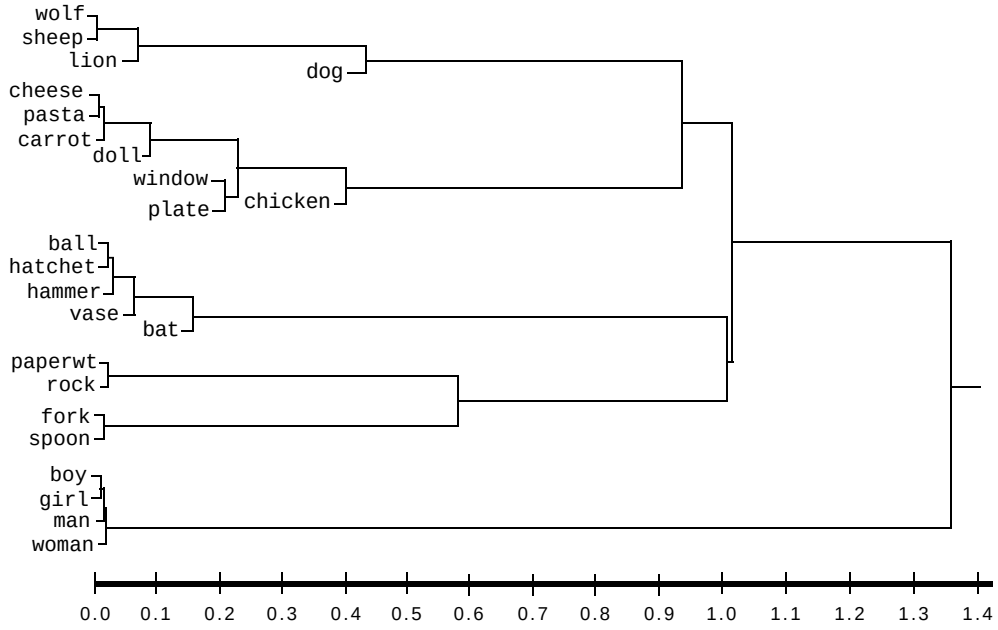

Figure 5: **Visualizing the similarities in the representations through merge clustering.** Step by step, the clusters with the shortest single linkage distance were merged. Distance is shown on the horizontal axis. The clustering follows the categories used in generating the sentences, showing that similar words have similar representations.

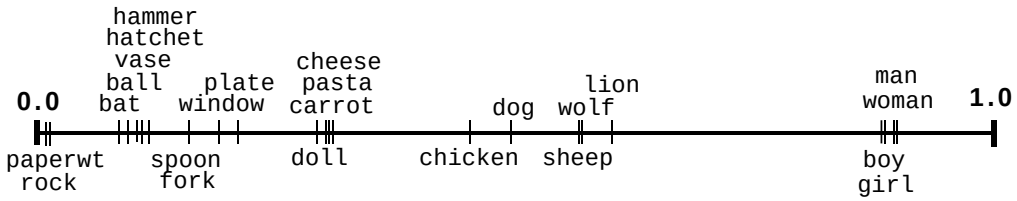

Figure 6: **Holographic representations.** The words are placed on a continuous line [0, 1] according to the value of the last component in their representations. The categorization is already evident in this single component.

representation can therefore be seen as a combination of 12 slightly different viewpoints into the word space. Such representations are very robust against errors. For example, one can eliminate units from the representation (by fixing them at 0.5 activation), and the decline in output accuracy is very gradual and approximately linear. Eliminating a unit means removing one classification perspective, and these perspectives apparently are additive.

Is it possible to name the properties the individual components are coding? The analysis of FGREP representations above suggests that the microfeatures in internal representations are identifiable only accidentally. Each individual component becomes sensitive to a combination of several features that is very unlikely to exactly match an established term, although partial matches are possible. For example, component 11 seems to separate animate objects, but the separation is not clear-cut and has a lot of finer structure (figure 6). Moreover, the network does not lose the information about animateness if this component becomes defective, so it would be incorrect to say that the component is responsible for representing "animateness" of the input data.

The case-role assignment network generalizes very well to new sentences. If an unfamiliar sentence is meaningful at all, its representation pattern is necessarily close to something the network has already seen. This is because FGREP develops similar representations for similarly behaving words. For example, the network has never seen `The man ate the chicken with a fork`, but its representation is very close to the familiar sentence, `The girl ate the pasta with a spoon`, because the representation for `girl` is equivalent to `man`, and `fork` to `spoon`, and `chicken` is very much like `pasta`. In more general terms, the system can process the word `x` in situation S, because it knows how to process the word `y` in situation S, and the words `x` and `y` are used similarly in a large number of other situations.

The representations code not only properties of the words themselves, but also the contexts in which they occur. As a result, the network generates expectations and guesses automatically. This is demonstrated in figure 3. During training, the network is required to produce the complete output pattern after each step in the sequence, even before it is possible to unambiguously recognize the sequence. As a result, the output patterns at each step are averages of all possible event sequences at that point, weighted by how often they have occurred. After the next word is input, some of the ambiguities are resolved and correct patterns are formed in the corresponding assemblies. Often the sentence representation is almost complete before the sentence has been fully input. In figure 3, the network has read `The boy hit the window` and has unambiguously assigned these words to the agent, act, and patient roles. The instrument and modifier slots indicate expectations. At this point it is already clear that the modifier slot is going to be blank, because only human patients have modifiers in the data. Most probably, an instrument is going to be mentioned later, and an expectation for a hitter (an average of all possible hitters) is displayed in the instrument slot.

If `with` is read next, a hitter is certain to follow, making the pattern even stronger. Reading a period next would instead clear the expectation in the instrument slot, indicating that the sentence is complete without this constituent. The expectations emerge automatically and cumulatively from the input word representations. Similarly to human language processing, the network can automatically fills in missing information, or select the correct sense for an ambiguous input word or guess meanings of unfamiliar words.

The properties discussed above make sybsymbolic representations very different from symbolic representations. Philosophically, the two approaches are incompatible. Because of the fundamentally different way of representing information, it is not possible to *exactly* duplicate the function of a symbolic system with the function of a subsymbolic system. A representation cannot be distributed and symbolic at the same time (van Gelder 1989, 1990). Consequently, subsymbolic representations are not just a low-level implementation of symbolic representations, but a fundamentally different approach to modeling natural language processing.

3 Modeling Human Language Processing

People seem to have two fundamentally different mechanisms at their disposal for performing cognitive tasks. Following a sequential symbolic strategy is the more obvious of the two. One does not have an immediate answer to the problem, but the answer is sequentially constructed from stored knowledge by a high-level goal-directed process, that is, by reasoning. Another type of cognitive processing occurs through associations immediately, in parallel, and without conscious control, in other words, by intuition. Large amounts of information, which may be incomplete or even conflicting, are simultaneously brought together to produce the most likely answer.

For cognitive processes based on conscious rule application, symbolic systems are a good approximation. However, intuitive processing cannot be easily implemented symbolically. In contrast, neural networks represent knowledge in terms of statistical regularities in their training, and processing is opaque, nonconcatenative, and immediate. Therefore, neural networks fit very well into modeling intuitive inference. A major issue is: are humans indeed symbol processors at the high level, for example in processing language, or could such processes be an epiphenomenon of low-level associative and statistical mechanisms?

3.1 Symbols vs. Soft Constraints

Processing sentences with embedded clauses is one task that shows how human language processing, although clearly symbolic at the surface level, at closer look exhibits strong subsymbolic qualities. Consider the following sentence:

```
The girl who the boy hit cried.
```

This sentence has a relative clause attached to the main noun `boy`. Relative clauses have a simple structure, and it is easy to form deeper embeddings by repeating the structure recursively:

```
The girl who the boy who the girl who lived next door blamed hit cried.
```

This sentence contains no new grammatical constructs. The familiar embedded clause construct is used just three times, and the resulting sentence is almost incomprehensible. If humans were truly

symbol processors, the number of levels would make no difference. It should be possible to handle each new embedding just like the previous one.

Now consider a similar sentence:

`The car that the man who the dog that had rabies bit drives is in the garage.`

This sentence has the same grammatical structure as the previous one, and for a symbol processor it should be equally easy, or hard, to process. Yet somehow this sentence is understandable, whereas the previous one was not. What is different?

Whereas the second sentence could be understood only based on syntactic analysis, the third one has strong semantic constraints between constituents. We know that dogs have rabies, people drive cars, and cars are in garages. These constraints make it possible for a human to understand the sentence even when they lose track of its syntactic structure. A symbol processor can parse the syntactic structure of both sentences perfectly well, and receives no benefit from the semantic constraints.

The conclusion from these examples is that people are not pure symbol processors when they understand language. Instead, all constraints—grammatical, semantic, discourse, pragmatic—are simultaneously taken into account to form the most likely interpretation of the sentence. This behavior is difficult to model with symbolic systems, but it is exactly what neural networks are good at. The example discussed in the next section illustrates how.

3.2 Example: Case-Role Assignment of Embedded Clauses

The SPEC system (figure 7; Miikkulainen 1996) is an extension of the subsymbolic parser architecture of section 2.2, designed to deal with sentences with embedded clauses. SPEC receives a sequence of word representations as its input, and for each clause in the sentence, forms an output representation indicating the assignment of words into case roles. The case-role representations are read off the system and placed in a short-term memory (currently outside SPEC) as soon as they are complete. SPEC consists of three main components: the Parser, the Segmenter, and the Stack. The Parser is a network similar to the parser above. The case-role assignment is represented at the output of the Parser as a case-role vector (CRV), that is, a concatenation of those three word representation vectors that fill the roles of agent, act, and patient in the sentence. For example, the word sequence `the girl saw the boy` receives the case-role assignment agent=`girl`, act=`saw`, patient=`boy`, which is represented as the vector `|girl saw boy|` at the output of the Parser network. When the sentence consists of multiple clauses, the relative pronouns are replaced by their referents: `The girl who liked the dog saw the boy` parses into two CRVs: `|girl liked dog|` and `|girl saw boy|`.

The Parser receives a continuous sequence of input word representations as its input, and its target pattern changes at each clause boundary. For example, in reading `The girl who liked the dog saw the boy` the target pattern representing `|girl saw boy|` is maintained during the first two words, then switched to `|girl liked dog|` during reading the embedded clause, and then back to `|girl saw boy|` for the rest of the sentence. The CRV for the embedded clause is read off the network after `dog` has been input, and the CRV for the main clause after the entire sentence has been read. When trained this way, the network is not limited to a fixed number of clauses by its output representation. Also, it does not have to maintain information about the entire past input sequence in its memory, making it possible in principle to generalize to new

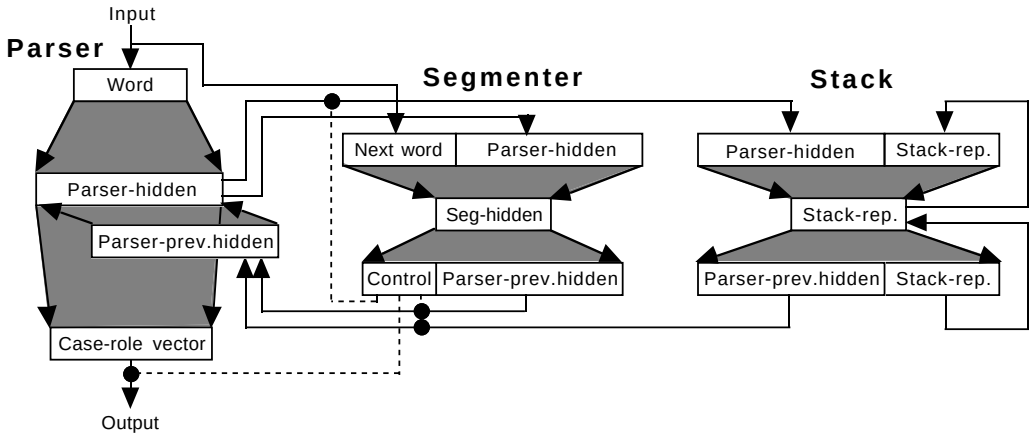

Figure 7: **Subsymbolic case-role assignment of sentences with embedded clauses.** The model, SPEC, consists of the Parser (a simple recurrent network), the Stack (a RAAM network), and the Segmenter (a feedforward network). The gray areas indicate propagation through weights, the solid lines stand for pattern transport, and the dashed lines represent control outputs (with gates).

clause structures. Unfortunately, after a center-embedding has been processed, it is difficult for the network to remember earlier constituents. This is why a Stack network is needed in SPEC.

The hidden layer of a Simple Recurrent Network forms a distributed representation of the sequence so far. The Stack has the task of storing this representation at each center embedding, and restoring it upon return from the embedding. For example, in parsing `The girl who liked the dog saw the boy`, the hidden-layer representation is pushed onto the stack after `The girl`, and popped back to the Parser's previous-hidden-layer assembly after `who liked the dog`. In effect, the SRN can then parse the top-level clause as if the center embedding had not been there at all.

The Stack is implemented as a Recursive Auto-Associative Memory (RAAM; Pollack 1990). RAAM is a three-layer backpropagation network trained to perform an identity mapping from input to output. As a side effect, the hidden layer learns to form compressed distributed representations of the network's input/output patterns, which consist of the top element of the stack and the distributed representation of the rest of the stack. The hidden layer representation is then recursively used as the representation for the rest of the stack in the next push operation. A potentially infinite stack can be compressed into a fixed-size representation this way. The stack structure can later be decoded by loading its distributed representation into the hidden layer and reading off the top element and the distributed representation for the rest of the stack at the output.

The Parser+Stack architecture alone is not quite sufficient for generalization into novel relative clause structures. For example, when trained with only examples of center embeddings (such as the above) and tail embeddings (like `The girl saw the boy who chased the cat`), the architecture generalizes well to new sentences such as `The girl who liked the dog saw the boy who chased the cat`. However, the system still fails to generalize to sentences like `The girl saw the boy who the dog who chased the cat bit`. Even though the Stack takes care of restoring the earlier state of the parse, the Parser has to learn all the different transitions into relative clauses. If it has encountered center embeddings only at the beginning of the sentence, it cannot generalize to a center embedding that occurs after an entire full clause has already been read.

This problem can be overcome with the Segmenter network. The Segmenter is trained to

recognize the transition to the relative clause, and to modify the hidden layer pattern so that only the relevant information remains (i.e `boy` in the above example). In other words, the controller has an internal representation for the "relative clause" construct, and applies it to change the hidden layer pattern so that the low-level sequence-processing network only has to deal with one type of clause boundary. By dividing the task into meta-level control, low-level pattern transformation, and memory, the whole system can generalize to novel structures.

SPEC was trained with 100 examples each of two sentence structures: (1) the two-level tail embedding (such as `The girl saw the boy who chased the cat who the dog bit`) and the two-level center-embedding (e.g. `the girl who the dog who chased the cat bit saw the boy`), and the stack was trained to store up to three levels of center embeddings. After training, SPEC generalized perfectly to all other combinations of center and tail embeddings of four clauses, that is, to 98,100 different sentences.

The main result, therefore, is that the SPEC architecture successfully generalizes not only to new instances of the familiar sentence structures, but to new structures as well. However, SPEC is not a mere reimplementation of a symbol processor. As SPEC's Stack becomes increasingly loaded, its output becomes less and less accurate; symbolic systems do not have any such inherent memory degradation. An important question is, does SPEC's performance degrade in a cognitively plausible manner, that is, does the system have similar difficulties in processing center embeddings as people do?

To elicit enough errors from SPEC to analyze its limitations, the Stack's performance was degraded by adding 30% noise in its propagation. Such an experiment can be claimed to simulate overload, stress, cognitive impairment, or lack of concentration situations. The system turned out to be remarkably robust against noise. The average Parser error rose somewhat, but the system still got 94% of its output words right. As expected, most of the errors occurred as a direct result of popping back from center embeddings with an inaccurate previous-hidden-layer representation. For example, in parsing `The girl, who the dog, who the boy, who chased the cat, liked, bit, saw the boy`, SPEC had trouble remembering the agents of `liked`, `bit` and `saw`, and patients of `liked` and `bit`. The performance depends on the level of the embedding in an interesting manner. It is harder for the network to remember the earlier constituents of shallower clauses than those of deeper clauses (figure 8). For example, SPEC could usually connect `boy` with `liked` (in 80% of the cases), but it was harder for it to remember that it was the `dog` who `bit` (58%) and even harder that the `girl` who `saw` (38%) in the above example.

Such behavior seems plausible in terms of human performance. Sentences with deep center embeddings are harder for people to remember than shallow ones (Foss and Cairns 1970; Miller and Isard 1964). It is easier to remember a constituent that occurred just recently in the sentence than one that occurred several embeddings ago. Interestingly, even though SPEC was especially designed to overcome such memory effects in the Parser's sequence memory, the same effect is generated by the Stack architecture. The latest embedding has noise added to it only once, whereas the earlier elements in the stack have been degraded multiple times. Therefore, the accuracy is a function of the number of pop operations instead of a function of the absolute level of the embedding.

When the SPEC output is analyzed word by word, several other interesting effects are revealed. Virtually in every case where SPEC made an error in popping an earlier agent or patient from the stack it confused it with another noun (54,556 times out of 54,603; random choice would yield 13,650). In other words, SPEC performs plausible role bindings: even if the exact agent or patient is obscured in the memory, it "knows" that it has to be a noun. Moreover, SPEC does not generate the noun at random. Out of all nouns it output incorrectly, 75% had occurred earlier in the

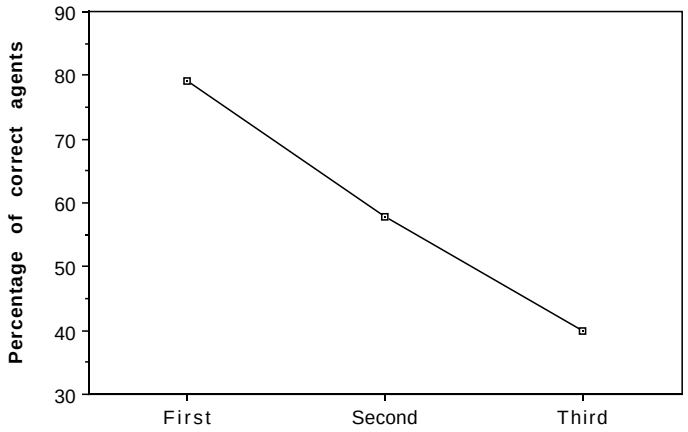

Figure 8: **Memory accuracy after return from center embeddings.** The percentage of correctly-remembered agents is plotted after the first, second, and the third pop in sentences with three levels of center embeddings. Each successive pop is harder and harder to do correctly. Similarly, SPEC remembers about 84% of the patients correctly after the first pop, and 67% after the second pop. The Stack representations were degrade with 30% noise to elicit the errors.

sentence, whereas a random choice would give only 54%. It seems that traces for the earlier nouns are discernible in the previous-hidden-layer pattern, and consequently, they are slightly favored at the output. Such priming effect is rather surprising, but it is very plausible in terms of human performance.

To test the effects of semantic constraints on sentence processing performance, the training sentences had beengenerated with a number of restrictions. A verb could have only certain nouns as its agent or patient. Some verbs had only one possible agent or patient, others had two, three or four. This way semantic restrictions of different strengths could be introduced in the training data. The restrictions turned out to have a marked effect on the performance (figure 9). If the agent or patient that needs to be popped from the stack is strongly correlated with the verb, it is easier for the network to remember it correctly. The effect depends on the strength of the semantic coupling. For example, `girl` is easier to remember in `The girl, who the dog bit, liked the boy`, than in `The girl, who the dog bit, saw the boy`, which is in turn easier than `The girl, who the dog bit, chased the cat`. The reason is that there are only two possible agents for `liked`, whereas there are three for `saw` and four for `chased`. While SPEC gets 95% of the unique agents right, it gets 76% of those with two alternatives, 69% of those with three, and only 67% of those with four.

A similar effect has been observed in human processing of relative clause structures. Half the subjects in Stolz's (1967) study could not decode complex center embeddings without semantic constraints. Huang (1983) showed that young children understand embedded clauses better when the constituents are semantically strongly coupled, and Caramazza and Zurif (1976) observed similar behavior in aphasics. This effect is often attributed to limited capability for processing syntax. The SPEC experiments indicate that it could be at least partly due to impaired memory as well. When the memory representation is impaired with noise, the Parser has to clean it up. In propagation through the Parser's weights, noise that does not coincide with the known alternatives cancels out. Apparently, when the verb is strongly correlated with some of the alternatives, more of the noise appears coincidental and is filtered out.

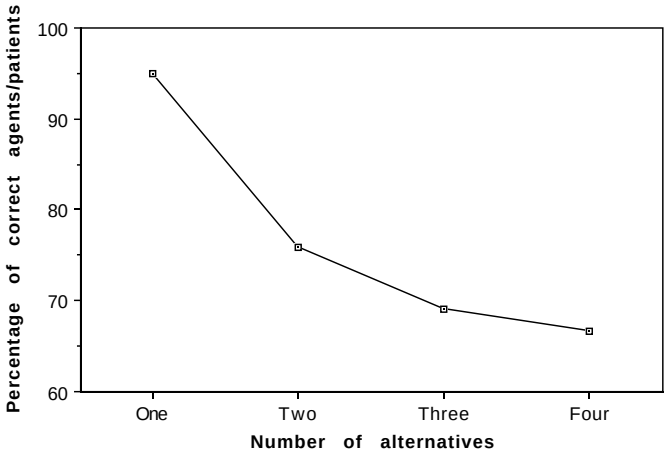

Figure 9: **Effect of the semantic restrictions on the memory accuracy.** The percentage of correctly-remembered agents and patients over the entire corpus is plotted against how strongly they were semantically associated with the verb. When there was only one alternative (such as `dog` as an agent for `bit` or `cat` as the patient of `chased`), SPEC remembered 95% of them correctly. There was a marked drop in accuracy with two, three and four alternatives. The Stack representations were degrade with 30% noise to elicit the errors.

The conclusion from the SPEC system is, then, that even when the subsymbolic architecture is designed for strong linguistic performance, such as generalizing to novel relative clause structures, it can exhibit subsymbolic effects similar to those of humans. Deep center embeddings are difficult for SPEC, and semantic constraints make the task easier. It is by tapping into such phenomena that the subsymbolic models can be most useful in natural language processing.

4 Overview of Subsymbolic Natural Language Processing

Natural language processing has been an active area of connectionist research for over a decade. Subsymbolic models have been developed to address a variety of issues, such as semantic interpretation, learning syntax and semantics, prepositional phrase attachment, anaphora resolution, morphology, phrase generation, identification of discourse topics, and goal-plan analysis (Allen 1987, 1989; Cosic and Munro 1988; Cottrell and Plunkett 1994; Gasser 1994; Karen 1990; Kukich 1987; Lee 1991; Munro et al. 1991; Touretzky 1991).

A good amount of work has been done showing that networks can capture grammatical structure. For example, Servan-Schreiber et al. (1991) showed how Simple Recurrent Networks can learn a finite state grammar. These networks are similar to those discussed above, except they are trained to predict the next item in the sequence instead of reading the sequence into a stationary representation. Servan-Schreiber et al. trained such an SRN with sample strings from a particular grammar, and it learned to indicate the possible next elements in the sequence. For example, given a sequence of distributed representations for elements B, T, X, X, V, and V, the network turns on two units representing X and S at its localist output layer, indicating that in this grammar, the string can continue with either X or S.

Elman (1991a, 1991b) used the same network architecture to predict a context-free language with embedded clauses. The network could not learn the language completely, but its performance

was remarkably similar to human performance. It learned better when it was trained incrementally, first with simple sentences and gradually including more and more complex examples. The network could maintain contingencies over embeddings if the number of intervening elements was small. However, deep center embeddings were difficult for the network, as they are for humans. Weckerly and Elman (1992) further showed that center embeddings were harder for this network than right-branching structures, and that processing was aided by semantic constraints between the lexical items. Such behavior matches human performance very well.

The above architectures demonstrated that subsymbolic networks build meaningful internal representations when exposed to examples of strings in a language. They did not address how such capabilities could be put to use in parsing and understanding language. McClelland and Kawamoto (1986) first identified the sentence case-role assigment as a good approach. The approach is particularly well-suited for neural networks because the cases can be conveniently represented as assemblies of units that hold distributed representations, and the parsing task becomes that of mapping between distributed representation patterns. McClelland and Kawamoto showed that given the syntactic role assignment of the sentence as the input, the network could assign the correct case roles for each constituent. The network also automatically performed semantic enrichment on the word representations (which were hand-coded concatenations of binary semantic features), and disambiguated between the different senses of ambiguous words.

As was discussed above in section 2.2, essentially the same task can be performed from sequential word-by-word input by a simple recurrent network and meaningful distributed representations for the words can be automatically developed at the same time. Systems with FGREP representations generally have a strong representation of context, which results in good generalization properties, robustness against noise and damage, and automatic "filling in" of missing information. The FGREP representations can be augmented with ID information, which allows the system to process a large vocabulary even after learning only a small number of distinct meanings (Miikkulainen and Dyer 1991). In this ID+content approach, representations for e.g. `John`, `Bill`, and `Mary` are created from the FGREP representation of `human` by concatenating unique ID patterns in front of it. All these words have the same meaning for the system, and it knows how to process them even if it has never seen them before.

St. John and McClelland (1990) further explored the subsymbolic approach to sentence interpretation in their Sentence Gestalt model. They aimed at explaining how syntactic, semantic, and thematic constraints are combined in sentence comprehension, and how this knowledge can be coded into the network by training it with queries. The gestalt is a hidden-layer representation of the whole sentence, built gradually from a sequence of input words by a simple recurrent network. The second part of the system (a three-layer backpropagation network) is trained to answer questions about the sentence gestalt, and in the process, useful thematic knowledge can be injected into the system. Similar approach can also be applied to processing script-based stories (St. John 1992).

A number of researchers have proposed modular and structured architectures. In addition to SPEC, FGREP networks were used to build a story processing system called DISCERN (Miikkulainen 1993). DISCERN is a large-scale natural language processing system implemented entirely at the subsymbolic level. DISCERN aims at bridging the gap between subsymbolic mechanisms and complex high-level behavior. Subsymbolic neural network models of parsing, generating, reasoning, lexical processing, and episodic memory are integrated into a single system that learns to read, paraphrase, and answer questions about stereotypical narratives. In this approach, connectionist networks are not only plausible models of isolated cognitive phenomena, but also serve as building

blocks for large-scale artificial intelligence systems.

In Jain's (1991) Structured Incremental Parser, one module was trained to assign words into phrases, and another to assign phrases into case roles. These modules were then replicated multiple times so that the recognition of each constituent was guaranteed independent of its position in the sentence. In the final system, words were input one at a time, and the output consisted of local representations for the possible assignments of words into phrases, phrases into clauses, phrases into roles in each clause, and for the possible relationships of the clauses. A consistent activation of the output units represented the interpretation of the sentence. The system could interpret complicated sentence structures, and even ungrammatical and incomplete input. The parse result was a description of the semantic relations of the constituents; the constituents themselves were not represented.

Berg's (1992) XERIC and Sharkey and Sharkey's (1992) parser were both based on the idea of combining a simple recurrent network with a Recursive Auto-Associative Memory (RAAM) that encodes and decodes parse trees. In Sharkey and Sharkey's model, first the RAAM network was trained to form compressed representations of syntactic parse trees. Second, an SRN network was trained to predict the next word in the sequence of words that make up the sentence. Third, a standard three-layer feedforward network was trained to map the SRN hidden-layer patterns into the RAAM parse-tree representations. During performance, a sequence of words was first read into the SRN, its final hidden layer transformed into a RAAM hidden layer, and then decoded into a parse tree with the RAAM network. Berg's XERIC worked in a similar manner, except the SRN hidden layer representations were directly decoded by the RAAM network.

Capabilities of RAAM networks and the distributed representations they form have been extensively studied (see e.g. Blank et al. 1992; Kwasny and Kalman 1995). Although the constituents of such representations, e.g. words in the parse tree, are not directly available, it is possible to perform "holistic" transformations on the entire patterns. For example, Chalmers (1990) trained one RAAM network to encode sentences in active voice, such as `John loves Michael`, and another RAAM network to encode same sentences in the passive, such as `Michael is loved by John`. A third, feedforward network was trained to map the distributed representation of the active sentence to the representation of the passive. The transformation network easily generalized to new sentences, showing that it had developed a sensitivity to structure that was only implicitly encoded in the distributed representations. Chrisman (1992) applied the same idea of holistic transformations to translating between English and Spanish sentences. Instead of two RAAMs and a transformation network he used two RAAMs with a common hidden layer. This forced the English and Spanish sentences to be encoded with more similar distributed representations, resulting in better performance.

The above results indicate that subsymbolic networks can represent and process linguistic knowledge in a cognitively valid manner, with strong sensitivity to context. They also show promise that complex structures can be processed, and large systems can be build from subsymbolic components.

5 Future Challenges

Even though the above systems are successful in what they are designed to do, and show very interesting cognitive behavior, they are still mostly demonstrations of capabilities on toy problems. Before subsymbolic natural language processing systems will rival the large symbolic NLP systems, several issues must be resolved:

How can complex linguistic representations be encoded on neural networks? For example, how can you represent an indefinite number of agents in a clause, or clauses in a sentence, or sentences in a story, when you only have a limited and fixed number of units in the network? People do not seem to have a fixed upper bound, although there clearly are memory limits. It is possible that some kind of reduced descriptions, similar to those modeled by RAAM, are being formed. However, so far it has turned out very difficult to make the RAAM architecture to generalize to new structures. For example in Sharkey and Sharkey's parser, and in Chalmers's and Chrisman's transformation systems, the limiting factor was the RAAM representation, not the transformation.

How can we come up with training examples for realistic language processing? Although large corpora of unprocessed text are readily available, subsymbolic systems usually require more sophisticated information as targets, such as case-role representations for parsing, or transfer examples for translation. Building such corpora is very laborous, and it is unclear whether it is ever possible to have large enough training sets, as long as generalization is limited to interpolation between training examples.

Building systems that would be able to generalize in a more fundamental way, by dynamic inferencing, or bringing together processing knowledge that has previously been seen in separate situations, is perhaps the greatest challenge for connectionist systems. It is only possible by strongly constraining the kinds of things the networks do, as in SPEC: the parser was limited to only simple pattern transformation, and the structure of the network forced generalization to novel structures. Time will tell how far such solutions will take us, but it is possible that network architectures and learning algorithms can be designed that would be able to learn metaknowledge about the kinds of tasks they are performing, and would allow them to use context when it is useful, and ignore it when it is irrelevant.

6 Conclusion

In this chapter, foundations for subsymbolic natural language processing were reviewed. Distributed representations were found to have very different properties from symbolic representations, properties that match the cognitive constraints in language processing very well. Two parsing architectures were discussed, one where properties of distributed representations could be easily illustrated, and another more complex architecture where the emergent cognitive effects could be analyzed. An overview of the state of the art in subsymbolic natural language processing systems suggests that many language phenomena can be modeled this way, although it is still difficult to scale the approach up to the level of complexity required by natural language processing in the real world.

Acknowledgements

This research was supported in part by the Texas Higher Education Coordinating Board under grant ARP-444.

Note

Source code for the FGREP, SPEC, and DISCERN systems, as well as an on-line demo of DISCERN, is available in the World Wide Web at URL http://www.cs.utexas.edu/users/nn.

References

Allen, R. B. (1987). Several studies on natural language and back-propagation. In *Proceedings of the IEEE First International Conference on Neural Networks* (San Diego, CA), vol. II, 335–341. Piscataway, NJ: IEEE.

Allen, R. B., and Riecken, M. E. (1989). Reference in connectionist language users. In Pfeifer, R., Schreter, Z., Fogelman Soulié, F., and Steels, L., editors, *Connectionism in Perspective*, 301–308. New York: Elsevier.

Berg, G. (1992). A connectionist parser with recursive sentence structure and lexical disambiguation. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 32–37. Cambridge, MA: MIT Press.

Blank, D. S., Meeden, L. A., and Marshall, J. B. (1992). Exploring the symbolic/subsymbolic continuum: A case study of RAAM. In Dinsmore, J., editor, *The Symbolic and Connectionist Paradigms: Closing the Gap*, 113–148. Hillsdale, NJ: Erlbaum.

Caramazza, A., and Zurif, E. B. (1976). Dissociation of algorithmic and heuristic processes in language comprehension: Evidence from aphasia. *Brain and Language*, 3:572–582.

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

Chrisman, L. (1992). Learning recursive distributed representations for holistic computation. *Connection Science*, 3:345–366.

Cook, W. A. (1989). *Case Grammar Theory*. Washington, DC: Georgetown University Press.

Cosic, C., and Munro, P. (1988). Learning to represent and understand locative prepositional phrases. In *Proceedings of the 10th Annual Conference of the Cognitive Science Society*, 257–262. Hillsdale, NJ: Erlbaum.

Cottrell, G. W., and Plunkett, K. (1994). Acquiring the mapping from meaning to sounds. *Connection Science*, 6:379–412.

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

Elman, J. L. (1991a). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–225.

Elman, J. L. (1991b). Incremental learning, or The importance of starting small. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, 443–448. Hillsdale, NJ: Erlbaum.

Fillmore, C. J. (1968). The case for case. In Bach, E., and Harms, R. T., editors, *Universals in Linguistic Theory*, 0–88. New York: Holt, Rinehart and Winston.

Foss, D. J., and Cairns, H. S. (1970). Some effects of memory limitation upon sentence comprehension and recall. *Journal of Verbal Learning and Verbal Behavior*, 9:541–547.

Gasser, M. (1994). Acquiring receptive morphology: A connectionist model. In *Proceedings of the 32nd Annual Meeting of the ACL*, 279–286.

Huang, M. S. (1983). A developmental study of children's comprehension of embedded sentences with and without semantic constraints. *Journal of Psychology*, 114:51–56.

Jain, A. N. (1991). Parsing complex sentences with structured connectionist networks. *Neural Computation*, 3:110–120.

Karen, L. F. R. (1990). Identification of topical entities in discourse: A connectionist approach to attentional mechanisms in language. *Connection Science*, 2:103–122.

Kukich, K. (1987). Where do phrases come from: Some preliminary experiments in connectionist phrase generation. In Kempen, G., editor, *Natural Language Generation: New Results in Artificial Intelligence, Psychology, and Linguistics*, 405–421. Dordrecht; Boston: Nijhoff.

Kwasny, S. C., and Kalman, B. L. (1995). Tail-recursive distributed representations and simple recurrent networks. *Connection Sciences*, 7:61–80.

Lee, G. (1991). *Distributed Semantic Representations for Goal/Plan Analysis of Narratives in a Connectionist Architecture*. PhD thesis, Computer Science Department, University of California, Los Angeles. Technical Report UCLA-AI-91-03.

McClelland, J. L., and Kawamoto, A. H. (1986). Mechanisms of sentence processing: Assigning roles to constituents. In McClelland, J. L., and Rumelhart, D. E., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2: Psychological and Biological Models*, 272–325. Cambridge, MA: MIT Press.

Miikkulainen, R. (1993). *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory*. Cambridge, MA: MIT Press.

Miikkulainen, R. (1996). Subsymbolic case-role analysis of sentences with embedded clauses. *Cognitive Science*, 20:47–73.

Miikkulainen, R., and Dyer, M. G. (1991). Natural language processing with modular neural networks and distributed lexicon. *Cognitive Science*, 15:343–399.

Miller, G. A., and Isard, S. (1964). Free recall of self-embedded English sentences. *Information and Control*, 7:292–303.

Munro, P., Cosic, C., and Tabasko, M. (1991). A network for encoding, decoding and translating locative prepositions. *Connection Science*, 3:225–240.

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

Servan-Schreiber, D., Cleeremans, A., and McClelland, J. L. (1991). Graded state machines: The representation of temporal contingencies in simple recurrent networks. *Machine Learning*, 7:161–194.

Sharkey, N. E., and Sharkey, A. J. C. (1992). A modular design for connectionist parsing. In Drossaers, M. F. J., and Nijholt, A., editors, *Twente Workshop on Language Technology 3: Connectionism and Natural Language Processing*, 87–96. Enschede, the Netherlands: Department of Computer Science, University of Twente.

St. John, M. F. (1992). The story gestalt: A model of knowledge-intensive processes in text comprehension. *Cognitive Science*, 16:271–306.

St. John, M. F., and McClelland, J. L. (1990). Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence*, 46:217–258.

Stolz, W. S. (1967). A study of the ability to decode grammatically novel sentences. *Journal of Verbal Learning and Verbal Behavior*, 6:867–873.

Touretzky, D. S. (1991). Connectionism and compositional semantics. In Barnden, J. A., and Pollack, J. B., editors, *High-Level Connectionist Models*, vol. 1 of *Advances in Connectionist and Neural Computation Theory,* Barnden, J. A., series editor, 17–31. Norwood, NJ: Ablex.

van Gelder, T. (1989). *Distributed Representation.* PhD thesis, Department of Philosophy, University of Pittsburgh, Pittsburgh, PA.

van Gelder, T. (1990). Compositionality: A connectionist variation on a classical theme. *Cognitive Science*, 14:355–384.

Weckerly, J., and Elman, J. L. (1992). A PDP approach to processing center-embedded sentences. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society*, 414–419. Hillsdale, NJ: Erlbaum.