# Disambiguation and Grammar as Emergent Soft Constraints

**Risto Miikkulainen and Marshall R. Mayberry III**
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
risto,martym@cs.utexas.edu

## Abstract

When reading a sentence such as "The diplomat threw the ball in the ballpark for the princess" our interpretation changes from a dance event to baseball and back to dance. Such on-line disambiguation happens automatically and appears to be based on dynamically combining the strengths of association between the keywords and the two senses. Subsymbolic neural networks are very good at modeling such behavior. They learn word meanings as soft constraints on interpretation, and dynamically combine these constraints to form the most likely interpretation. On the other hand, it is very difficult to show how systematic language structures such as relative clauses could be processed in such a system. The network would only learn to associate them to specific contexts and would not be able to process new combinations of them. A closer look at understanding embedded clauses shows that humans are not very systematic in processing grammatical structures either. For example, "The girl who the boy who the girl who lived next door blamed hit cried" is very difficult to understand, whereas "The car that the man who the dog that had rabies bit drives is in the garage" is not. This difference emerges from the same semantic constraints that are at work in the disambiguation task. In this chapter we will show how the subsymbolic parser can be combined with high-level control that allows the system to process novel combinations of relative clauses systematically, while still being sensitive to the semantic constraints.

## 1 Introduction

How do people arrive at an interpretation of a sentence? The thesis put forth by this chapter is that sentence processing is not a crisp symbolic process, but instead emerges from bringing together a number of soft constraints, such as the syntactic structure of the sentence and semantic associations between its constituents. 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 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.`

To appear in B. J. MacWhinney (editor), *Emergentist Approaches to Language*, Hillsdale, NJ: Erlbaum.

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. The reason is that the previous sentence can only be understood based on syntactic analysis, whereas this 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.

These examples suggest 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. What could be the underlying mechanism for representing and combining such constraints?

A viable hypothesis is that the constraints arise from correlations with past experience. For example, if the sentence understander has often heard about dogs with rabies, rarely about people with rabies, and never about cars with rabies, he or she has a semantic constraint that favors attaching rabies to dog in the above sentence. It is not a hard rule, since it might also be that the man had rabies, but it is a soft constraint that makes those interpretations of the sentence where dog has rabies more prominent. When several such constraints are simultaneously taken into account, what results is the most likely interpretation of the sentence, given the past experience.

The theory of sentence processing as soft constraints can be made concrete in subsymbolic neural network models. Processing in neural networks is based on precisely the same principles. Trained to associate a set of input patterns to a set of output patterns, the network learns correlations in the data. For each new input pattern, the network then forms the most likely output given the training data (Hertz et al. 1991). If the sentence interpretation task can be cast as a mapping from a sequence of input words to a representation of the meaning of the sentence, sentence processing can be modeled with neural networks. Such models can give us insight into the processes underlying sentence processing, and suggest further hypotheses that can be tested experimentally.

Several subsymbolic models of sentence processing have been built around this idea (Elman 1991; McClelland and Kawamoto 1986; Miikkulainen and Dyer 1991; St. John and McClelland 1990). This chapter aims at two goals: (1) providing a clear, concrete example of soft constraints and how they are combined in sentence processing, and (2) showing how systems based on soft constraints can process language productively and systematically, while at the same time maintaining cognitive validity (as in the examples above).

## 2 Disambiguation as Soft Constraints

In understanding ambiguous sentences, humans seem to employ automatic and immediate lexical disambiguation mechanisms even when they are compelled to alternate between two or more senses of an ambiguous word. Consider the sentence

`The diplomat threw the ball in the ballpark for the princess.`

As a reader processes this sentence, he or she is inclined to interpret the word `ball` first as a dance event, then as baseball, and lastly again as dance. Yet this processing seems to occur at such a low level that the reader will hardly be aware that there was any conflict in his interpretation. There does not seem to be any conscious inferencing required, no moment's cogitation as might be observed if, say, the reader were instead asked to compute the product of two double-digit numbers.

Several models have been proposed to account for how ambiguities are resolved during reading. The three most prominent in recent years have been the context-dependent, the single-access, and the multiple-access model. The context-dependent model (Glucksberg et al. 1986; Schvaneveldt et al. 1976) is based on the assumption that only one meaning of a word is activated at any given time, namely the one most appropriate to the context in which the word occurs. The primary reason is that the context primes the meaning which is most applicable, while suppressing others. The single access (or ordered-access) model (Forster and Bednall 1976; Hogaboam and Perfetti 1975; Simpson and Burgess 1985) posits that only one active interpretation of an ambiguous sentence is maintained at any one time. If in the course of processing the sentence information is encountered that does not accord well with the active interpretation, then that interpretation is abandoned and a representation that accounts for the established information as well as for the current ambiguity is sought, most probably through backtracking to the point of ambiguity. The activation level of an interpretation is determined by the relative frequencies of the meanings of the word or words that are the source of the ambiguity. The search process for the appropriate meaning takes place serially, terminating when a fit is made, or retaining the most dominant meaning when no contextually relevant match can be found. In the strongest statement of the model (Hogaboam and Perfetti 1975), only the most dominant meaning of an ambiguous word is retrieved first, regardless of whether the context supports a subordinate meaning.

The multiple access model (Onifer and Swinney 1981; Seidenberg et al. 1982; Tanenhaus et al. 1979), which is the most widely accepted, suggests that several interpretations may be actively maintained when ambiguous information is encountered. At a later time, when additional input allows resolving the ambiguity, only the appropriate interpretation is retained. However, not all of the interpretations may be maintained with equal activation levels. Rather, the strength of a particular activation would be proportional to the likelihood of that interpretation being the correct one. Unlike the single access model, in which a single meaning is sought and selected, the multiple access model claims that all meanings are activated simultaneously regardless of context, but the context later influences selection of the most appropriate one. As is not unusual when aspects of behavior are supported by several more or less opposing models, refinements are proposed to include elements from several models. For example, Burgess and Simpson (1988) support the multiple access model, but suggest that meaning frequencies determine which interpretations reach the recognition threshold first. The role of context is to select which of the meanings remains activated.

This is where a subsymbolic neural neural model can be used to gain insight into the process. It is possible to build a model of sentence understanding where ambiguity resolution occurs as an integral part of the task. Observing the behavior of the network, we see that multiple activation levels are maintained simultaneously during the processing of a sentence, and the various meanings of each word are activated to the degree that corresponds to the frequency with which that word has been associated to the previous words in the sentence in the past. This way, the model confirms Burgess and Simpson's theory computationally, and suggests that semantic frequencies could play an even more fundamental role: lexical ambiguity resolution emerges from combining the soft constraints posited by the semantic frequencies of the sentence constituents.

## 2.1 The Sentence Disambiguation Model

The sentence processing model is based on the Simple Recurrent Network architecture (SRN; Elman 1990), which has recently become a standard tool in subsymbolic natural language processing, including sentence and story understanding (Miikkulainen 1993, 1996; St. John and McClelland 1990; St. John 1992) and learning grammatical structure (Elman 1991; Servan-Schreiber et al. 1991). In modeling disambiguation, the idea is to train the SRN to map a sequence of words to the case-role representation (Fillmore 1968; Cook 1989) of the sentence, and observe the evolution of the sentence representation during parsing an ambiguous sentence. The parser was trained on sentences generated from two basic sentence templates:

1. `The` *agent* `threw the ball in the` *location* `for the` *recipient.*
2. `The ball was thrown in the` *location* `for the` *recipient* `by the` *agent.*

The fixed words in the template are indicated by `courier` typeface. The *location*, *recipient*, and *agent* stand for slots to be filled by actual content words. Depending on these words, the sentences could be interpreted as statements about baseball (e.g. `The pitcher threw the ball in the ballbark for the fans`), dance (`The emcee threw the ball in the ballroom for the princess`) or something rather ambiguous (`The visitor threw the ball in the court for the victor`). The output of the parser is one of two possible case-role representations:

1. *agent* *act:*`tossed` *patient:*`baseball` *location* *recipient*

| **Feature** | Set to 1 for words |
|---|---|
| **1. Ball** | `ball`, `baseball` and `dance` |
| **2. Verb** | `thrown`, `tossed` and `hosted` |
| **3. Other** | `the` and `was` |
| **4. Preposition** | `in`, `for`, and `by` |
| **5. Location** | the five *location* words and `in` |
| **6. Recipient** | the five *recipient* words and `for` |
| **7. Agent** | the five *agent* words and `by` |
| **8. Sense** | Graduated according to word sense |

Table 1: **The word representation vectors.** The words in the lexicon were encoded by these eight features. The first seven components were set to either 0 or 1; the right column lists those words that had the value 1. The **Sense** feature was used to indicate the degree of association to which a word had the two senses of `throw` and `ball`.

| Location | Recipient | Agent | Sense |
|---|---|---|---|
| `ballpark` | `fans` | `pitcher` | 0.00 |
| `stadium` | `press` | `coach` | 0.25 |
| `court` | `victor` | `visitor` | 0.50 |
| `clubroom` | `vips` | `diplomat` | 0.75 |
| `ballroom` | `princess` | `emcee` | 1.00 |

Table 2: **Sense values for the content words.**

| Fixed words | Sense |
|---|---|
| `tossed, baseball` (output only) | 0.00 |
| `ball,thrown/threw,the,was,in,for,by` | 0.50 |
| `hosted, dance` (output only) | 1.00 |

Table 3: **Sense values for the fixed words.**

2. *agent act:*`hosted` *patient:*`dance` *location recipient.*

Which of these two representations were used in the output depended on how strongly the words occupying the *location*, *recipient*, and *agent* slots were associated with baseball and dance.

### 2.1.1 Training data

There were a total of 26 words in the lexicon: five for each of the three slots, two interpretations of `throw` and `ball`, and seven fixed words for the input sentences. Each were given hand-coded representation vectors according to the eight features shown in table 1. The last component, **sense**, indicates how strongly the word is associated with `tossed baseball` (0) and `hosted dance` (1). For example, if a word representation has a sense of 0.25, it is more strongly associated with baseball than dance. Tables 2 and 3 summarize the sense values of each word.

This representation strategy was chosen because it makes the disambiguation process transparent. To see what the network is thinking, it is enough to look at the values of the sense units at its output. Distributed word representations (e.g. Miikkulainen 1993) could be used as well, but sense would then have to be calculated based on distances between representation vectors.

The sense values were then used to put together a set of training sentences that made the sense associations explicit. For each possible sentence, the sense was computed by averaging the sense values of the individual words. Since these values were graduated in fourths, averaging over the three content words (per sentence) would result in twelfths. Thus, each input sentence was repeated twelve times in the training corpus, with the two possible case-role representations (i.e. senses) assigned in proportion of the sense value of the sentence. For example, if the passive sentence template is instantiated with the words `clubroom` (sense: 0.75), `fans` (0.00), and `emcee` (1.00), the following sentences is obtained:

```
The ball was thrown in the clubroom for the fans by the emcee.
```

Averaging the sense values gives $\frac{7}{12}$, or 0.5833. Accordingly, this sentence was repeated twelve times in the training corpus, with seven of the sentences assigned to `hosted dance` at the output, and five to `tossed baseball`. Thus, in the experience of the parser, $58\frac{1}{3}\%$ of the contexts in which `ball`, `clubroom`, `fans`, and `emcee` appeared were associated with `hosted dance`, and the remaining with `tossed baseball`. Hence, the dance sense would be slightly more dominant in this context, and would be the preferred interpretation.

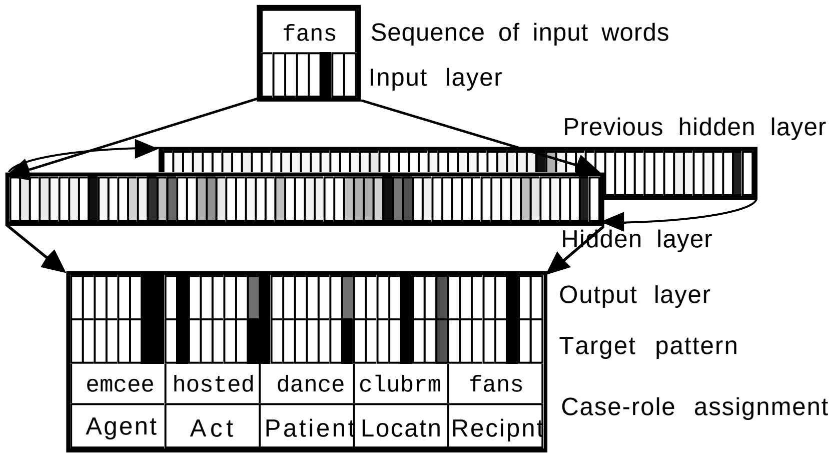

Figure 1: **The simple recurrent parser architecture.** The model consists of a simple recurrent network trained through backpropagation to map a sequence of input word representations into a case-role representation of the sentence.

There are 125 possible combinations of the five words in the three categories. Each combination was used to instantiate the two sentence templates, giving a total of 250 sentences. Since each sentence is repeated 12 times, the training corpus is composed of 3000 sentences. These sentences comprise the contextual history of the ambiguous words `throw` and `ball`. Active and passive constructions were used to contrast whatever priming effects the words might have on the general sense of the sentence; the fact that they represent different voices is not significant in this experiment.

### 2.1.2 Network architecture

The SRN parser network (figure 1) has a single input assembly consisting of eight units, corresponding to the eight components in the word representation. The output layer is a concatenation of five word-representation assemblies, corresponding to the case-role assignment of the sentence.

At each step in the sequence, a word representation is loaded in the input assembly and the activity is propagated through the hidden layer to the output. The activity in the hidden layer (60 units wide) is saved in the previous-hidden-layer assembly, and used together with the word representation as input to the hidden layer in the next step. Throughout the sequence, the complete case-role assignment is used as the training target, and the error is propagated and the weights are changed (through the backpropagation algorithm) at each step.

In effect, the network is trained to shoot for the complete sentence interpretation from the first word on. As a result, it learns to indicate the current sense of the sentence in the **sense** components of the *act* and the *patient* assemblies at its output. If the current interpretation is predominantly `hosted dance`, these components have high values, and if it is `tossed baseball`, they have low values. A completely ambiguous interpretation is indicated by activation 0.5.

In this experiment, the parser was trained with a 0.5 learning rate for 100 epochs, then 0.1 for 50 epochs, 0.05 for 5, and finally 0.01 until epoch 200. At this point, the average error per unit after reading a complete sentence was 0.024. The exact shape of the training schedule is not crucial: as long as the learning rate is gradually decreased, good results can be achieved reasonably quickly and reliably.

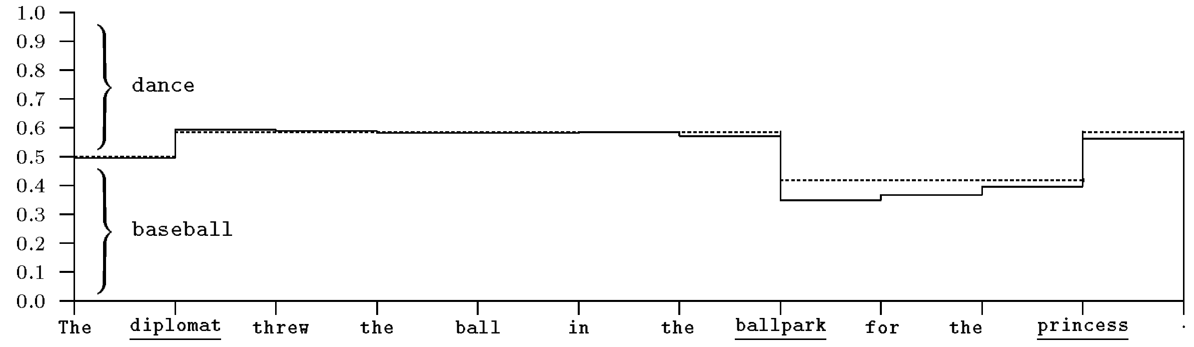

Figure 2: **Evolution of the interpretation of an active construction.** The dotted line represents the theoretical sense level during processing the sentence, and the solid line indicates the average of the two **sense** output units. The content words have been underlined. The average error per unit on this sentence was 0.0180.

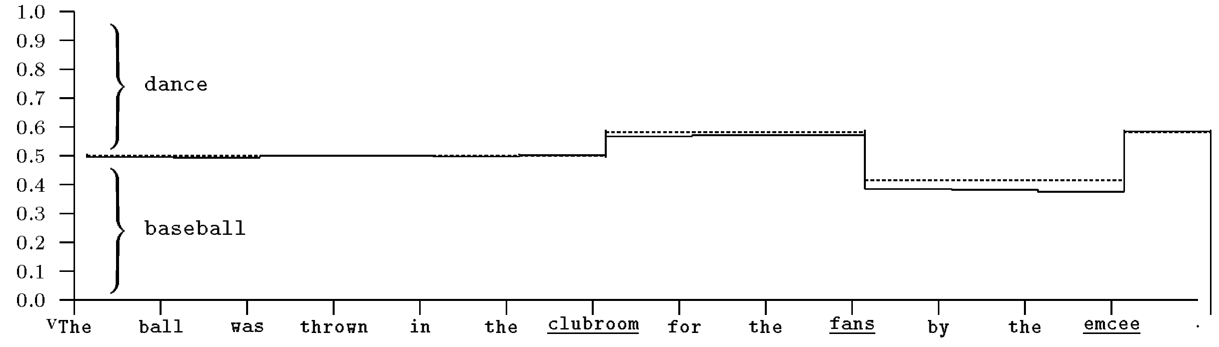

Figure 3: **Interpretation of a passive construction.** The average error per unit on this sentence was 0.0123.

## 2.2 Disambiguation Results

The parser was tested with the same set of sentences used to train it to determine how well it captured the sense for each sentence. The theoretically optimal values for the sense outputs were obtained from the training data based on the distribution of the words in the sentences. The **sense** outputs of the network were then compared to the theoretical values. Generalization was not tested in this study because tight control over the theoretical frequencies was desired, and because good generalization is common for this type of models and offers no new perspective on the disambiguation problem.

The network had captured the sense frequencies very accurately: The average error across the entire data set was found to be 0.0114 (0.0122 for the active constructions and 0.0107 for the passive). Moreover, all sentences where at least one of the content words was associated to a sense opposite of that of the other content words resulted in semantic flipping behavior. Below, the processing of two sentences, one active and one passive, is analyzed in detail. These examples are particularly interesting because they require revising the semantic interpretation twice during processing.

In reading the active sentence

```
The diplomat threw the ball in the ballpark for the princess
```

(figure 2), the average of the two sense unit activations is initially very nearly 0.5, indicating no bias one way or the other (i.e. complete ambiguity) because the two senses of `ball` were equiprobable in the training set. After processing the word `diplomat`, the activation level rises to 0.5921 since a slight majority ($58\frac{1}{3}\%$) of the sentences in the training set in which `diplomat` occurs have the sense `dance`. In effect, `diplomat` has a priming effect on the interpretation of the rest of the words. The activation remains at roughly this level until the word `ballpark` is encountered. At this point, the semantic bias *flips* to 0.3481 in favor of `baseball`, because in the experience of the parser, a majority ($58\frac{1}{3}\%$) of the sentences in which both `diplomat` and `ballpark` appear have the sense of `baseball`. The activation stays below 0.5 until `princess` is read in as the last word, whereupon it flips back to 0.5610, indicating that the sentence is once again interpreted as `diplomat hosted dance`. The theoretical expectation of those sentences containing the words `diplomat`, `ballpark` and `princess` is 0.5833, which is close to the activation level the parser finally settled upon.

Similarly, in processing the passive sentence

```
The ball was thrown in the clubroom for the fans by the emcee
```

(figure 3), after a long sequence of neutral fixed words, the network encounters `clubroom` and the interpretation becomes biased toward `dance`, because $58\frac{1}{3}\%$ of the training sentences with `clubroom` have this sense. Upon reading `fans`, the interpretation flips toward `baseball`, because now the majority of sentences (again $58\frac{1}{3}\%$) with both `clubroom` and `fans` have the sense `baseball`. When the last word is read, the bias flips again back to `dance`, because a sentence with `clubroom, fans` and `emcee` has an overall sense average 0.5833.

The biases and flips in the sense values are not particularly dramatic because the frequency differences are small in the training corpus. For a more stark contrast, these allocations could be adjusted; however, it is important that even such minor differences will result in reliable semantic revision behavior. In real world data there is a lot of noise in terms of accidental correlations, originating from many simultaneous tasks and goals. Even if the semantic regularities are only slightly stronger than the noise, the system will be able to utilize them to perform semantic interpretation reliably.

## 2.3 Conclusions from the Sentence Disambiguation Model

The most salient effect was that the semantic sense of an input sentence as a whole varied as a function of the semantic senses of its component words. It is this variation that accounts for the flipping behavior that we set out to model. Let us examine what insights the model can provide into human language comprehension.

A reader has experienced each word in a variety of contexts. Instead of regarding the word semantics as a collection of discrete and disjoint definitions in the lexicon, it is possible to view semantics simply as an encoding of all these past contexts. For most words, these contexts share much in common. For an ambiguous word, however, there are two or more distinctly different contexts, some of them more frequently observed than others. As a reader processes a sentence, there is an interaction between the semantics (i.e. past contexts) of each word and the evolving interpretation of the current sentence. Each word primes the interpretation according to the frequency with which the word has been associated with the current context in the past. After reading the whole sentence, all the past contexts of its constituent words are combined into the most likely interpretation. In other words, the interpretation emerges from the soft constraints among the sentence constituents.

Although sense disambiguation is just one small part of language processing, the same mechanisms could in principle be at work in language processing in general. Syntactic, semantic, discourse, and pragmatic structures manifest themselves as regularities in how language is used. These regularities can be captured and represented in the mind—and modified continuously with experience—to make language comprehension possible.

This approach is sufficient for capturing the statistical, automatic aspect of sentence comprehension, where the meaning of the sentence can be derived by combining prior contexts. However, it is not sufficient for explaining how dynamic inferencing comes about, where constraints that have previously been seen only in separate contexts must be combined. For example, in the sentence (due to Lange and Dyer 1989)

```
John put the pot in the dishwasher because the police were coming,
```

if dishwasher and police have never been seen together before, there is no basis for statistically combining the contexts they represent. Washing dishes and waiting for the police to come does not combine into using a dishwasher to hide things. Such cases require a high-level process that has access to the statistical semantics,

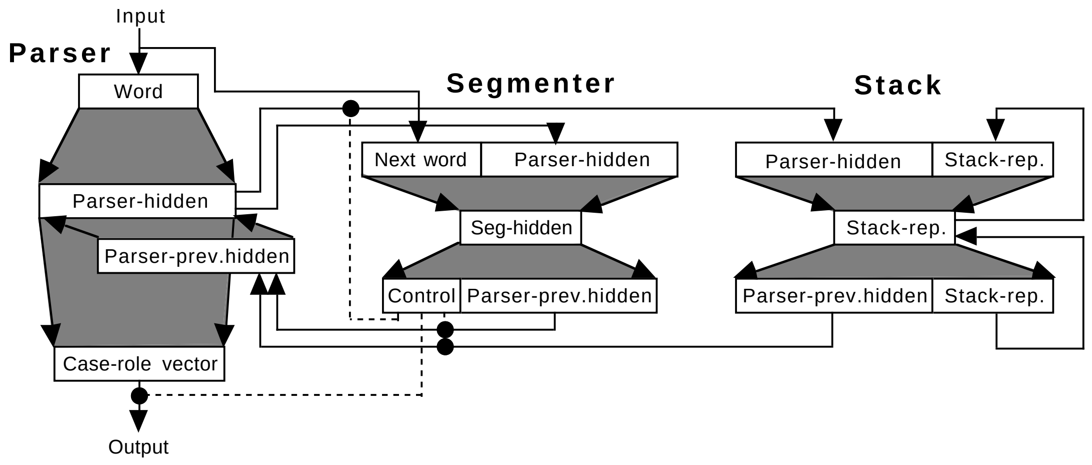

Figure 4: **The SPEC sentence processing architecture.** The system consists of the Parser (a simple recurrent network), the Stack (a RAAM network), and the Segmenter (a feedforward network). The gray areas indicate propagation through weights, the solid lines stand for pattern transport, and the dashed lines represent control outputs (with gates).

but can use higher-level constraints to combine constituents into novel representations. An approach and an example of such a process is described next.

## 3 Processing Grammatical Structure

The problem of dynamic inferencing is particularly obvious in processing sentences with relative clauses. A subsymbolic neural network can be trained to form a case-role representation of each clause in a sentence like `The girl who liked the dog saw the boy`, and it will generalize to different versions of the same structure, such as `The dog who bit the girl chased the cat` (Miikkulainen 1990). However, such a network cannot parse sentences with novel combinations of relative clauses, such as `The girl who liked the dog saw the boy who chased the cat`. This sentence has a relative clause in a new place in an otherwise familiar sentence structure, and processing it would require combining contexts that the network has never seen together before. Such a lack of generality is a serious problem, given how effortlessly people can understand novel sentences.

This section will demonstrate that dynamic inferencing is possible if the soft constraints are combined with high-level control. In the SPEC model (Subsymbolic Parser for Embedded Clauses; Miikkulainen 1996), the basic idea is to separate the tasks of segmenting the input word sequence into clauses, forming the case-role representations, and keeping track of the recursive embeddings into different modules. Each module is trained with only the most basic relative clause constructs, and the combined system is able to generalize to novel sentences with remarkably complex structure. Importantly, SPEC is not a neural network reimplementation of a symbol processor. It is a self-contained, purely subsymbolic neural network system, and exhibits the usual properties of such systems. For example, unlike symbolic parsers, the network exhibits plausible memory degradation as the depth of the center embeddings increases, its memory is primed by the earlier constituents in the sentence, and its performance is aided by semantic constraints between the constituents.

### 3.1 The SPEC Architecture

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 (figure 4).

### 3.1.1 The Parser

The Parser is similar to the disambiguation network of section 2. It performs the actual transformation of the word sequence into the case-role representations. Words are represented distributively as vectors of gray-scale values between 0 and 1. The component values are initially assigned randomly and modified during learning by the FGREP method (Miikkulainen and Dyer 1991; Miikkulainen 1993) so that similar words will have similar representations. FGREP is a convenient way for forming these representations, but SPEC is not dependent on FGREP. The word representations could have been obtained through semantic feature encoding (McClelland and Kawamoto 1986) as well, or even assigned randomly.

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 (the representation was limited to three roles for simplicity). 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 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.

### 3.1.2 The Stack

The hidden layer of a simple recurrent network forms a compressed description 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 network is a model of one specific task of the human working memory. Although the working memory must be responsible for several different functions, in SPEC it is only required to perform the function of the symbolic stack; hence, it is called the Stack network. It would of course be possible to just use a symbolic stack in SPEC, but as we will see in section 3.2, many of the cognitive effects in SPEC depend on subsymbolic representations of memory traces.

There are many ways to form such memory traces: for example, in the hidden layer of an SRN, or as attractors in a recurrent network (Hinton and Shallice 1991; Plaut 1991). From the point of view of SPEC, it is only important that the subsymbolic representation of the current state is a combined representation of the previous state and the new item. The most direct architecture for this task is the 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, its hidden layer learns to form compressed representations of the network's input/output patterns. These representations can be recursively used as constituents in other input patterns, and a potentially infinite hierarchical data structure, such as a stack, can this way be compressed into a fixed-size representation.

The input/output of the Stack consists of the stack's top element and the compressed representation for the rest of the stack. Initially the stack is empty, which is represented by setting all units in the "Stack-rep" assembly to 0.5 (figure 4). The first element, such as the hidden-layer pattern of the Parser network after reading `The girl`, is loaded into the "Parser-hidden" assembly, and the activity is propagated to the hidden layer. The hidden-layer pattern is then loaded into the "Stack-rep" assembly at the input, and the Stack network is ready for another push operation.

When the Parser returns from the center embedding, the stored pattern needs to be popped from the stack. The current stack representation is loaded into the hidden layer, and the activity is propagated to the output layer. At the output, the "Parser-prev.hidden" assembly contains the stored Parser-hidden-layer pattern, which is then loaded into the previous-hidden-layer assembly of the Parser network (figure 4). The "Stack-rep" assembly contains the compressed representation for the rest of the stack, and it is loaded to the hidden layer of the Stack network, which is then ready for another pop operation.

### 3.1.3 The Segmenter

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.

The solution is to train an additional network, the Segmenter, to divide the input sequence into clauses. The segmenter receives the current hidden-layer pattern as its input, together with the representation for the next input word, and it is trained to produce a modified hidden-layer pattern as its output (figure 4). The output is then loaded into the previous-hidden-layer assembly of the Parser. In the middle of reading a clause, the Segmenter passes the hidden-layer pattern through without modification. However, if the next word is a relative pronoun, the segmenter modifies the pattern so that only the relevant information remains. In the above example, after `boy` has been read and `who` is next to come, the Segmenter generates a pattern similar to that of the Parser's hidden layer after only `The boy` in the beginning of the sentence has been input.

In other words, the Segmenter (1) detects transitions to relative clauses, and (2) changes the sequence memory so that the Parser only has to deal with one type of clause boundary. This way, the Parser's task becomes sufficiently simple so that the entire system can generalize to new structures.

The Segmenter plays a central role in the architecture, and it is very natural to give it a complete control over the entire parsing process. Control is implemented through three additional units at the Segmenter's output (figure 4). The units "Push" and "Pop" control the stack operations, and the unit "Output" indicates when the Parser output is complete and should be read off the system. This way the Segmenter implements the high-level strategy of parsing sentences with relative clauses.

Although the Segmenter was originally found necessary for computational reasons (i.e. for generalization to new sentence structures), it can also be motivated in a more general cognitive framework of perspective shifts in processing relative clauses (MacWhinney 1977, 1988). When a relative pronoun is encountered, a perspective shift is initiated: The current clause is temporarily suspended, and a new clause is started with possibly new agent or patient, introducing a new perspective. The Segmenter is a mechanism for carrying out this shift. It has a global view of the parsing process, allowing it to decide when a shift is necessary. It then utilizes the low-level mechanisms of working memory of the Stack and the sequence memory of the Parser to carry it out.

| | | |
|---|---|---|
| S | → | NP VP |
| NP | → | DET N \| DET N RC |
| VP | → | V NP |
| RC | → | who VP \| who NP V |
| N | → | boy \| girl \| dog \| cat |
| V | → | chased \| liked \| saw \| bit |
| DET | → | the |

Table 4: **The sentence grammar.**

| Verb | Case-role | Possible fillers |
|---|---|---|
| chased | Agent: | boy,girl,dog,cat |
| | Patient: | cat |
| liked | Agent: | boy,girl |
| | Patient: | boy,girl,dog |
| saw | Agent: | boy,girl,cat |
| | Patient: | boy,girl |
| bit | Agent: | dog |
| | Patient: | boy,girl,dog,cat |

Table 5: **Semantic restrictions.**

## 3.2 Experiments

The training and testing corpus was generated from a simple phrase structure grammar (table 4). Each clause consisted of three constituents: the agent, the verb and the patient. A relative `who`-clause could be attached to the agent or to the patient of the parent clause, and `who` could fill the role of either the agent or the patient in the relative clause. In addition to `who` and `the`, the vocabulary consisted of the verbs `chased`, `liked`, `saw` and `bit`, and the nouns `boy`, `girl`, `dog` and `cat`. Certain semantic restrictions were imposed on the sentences. A verb could only have certain nouns as its agent and patient, as listed in table 5. The grammar was used to generate all sentences with up to four clauses, and those that did not match the semantic restrictions were discarded. The final corpus consisted of 49 different sentence structures, with a total of 98,100 different sentences.

The SPEC architecture divides the sentence parsing task into three subtasks. Each component needs to learn only the basic constructs in its task, and the combined architecture forces generalization into novel combinations of these constructs. Therefore, it is enough to train SPEC with only 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`). The training set consisted of 100 randomly-selected sentences of each type. In addition, the Stack was trained to encode and decode up to three levels of pushes and pops.

The word representations consisted of 12 units. Parser's hidden layer was 75 units wide, Segmenter's 50 units, and Stack's 50 units. All networks were trained with on-line backpropagation with 0.1 learning rate and without momentum. Both the Parser and the Segmenter developed word representations at their input layers (with a learning rate of 0.001). The networks were trained separately (i.e. without propagation between modules) and simultaneously, sharing the same gradually-developing word and parser-hidden-layer representations. The convergence was very strong. After 400 epochs, the average error per output unit was 0.018 for the Parser, 0.008 for the Segmenter (0.002 for the control outputs), and 0.003 for the Stack.

SPEC's performance was then tested on the entire corpus of 98,100 sentences. The patterns in the Parser's output assemblies were labeled according to the nearest representation in the lexicon. The control output was taken to be correct if those control units that should have been active at 1 had an activation level greater than 0.7, and those that should have been 0 had activation less than 0.3. Measured this way, the performance was excellent: SPEC did not make a single mistake in the entire corpus, neither in the output words or in control. The average unit error was 0.034 for the Parser, 0.009 for the Segmenter (0.003 for control), and 0.005 for the Stack. There was very little variation between sentences and words within each sentence, indicating that the system was operating within a safe margin.

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, thereby demonstrating dynamic inferencing. 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 recursive structures 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

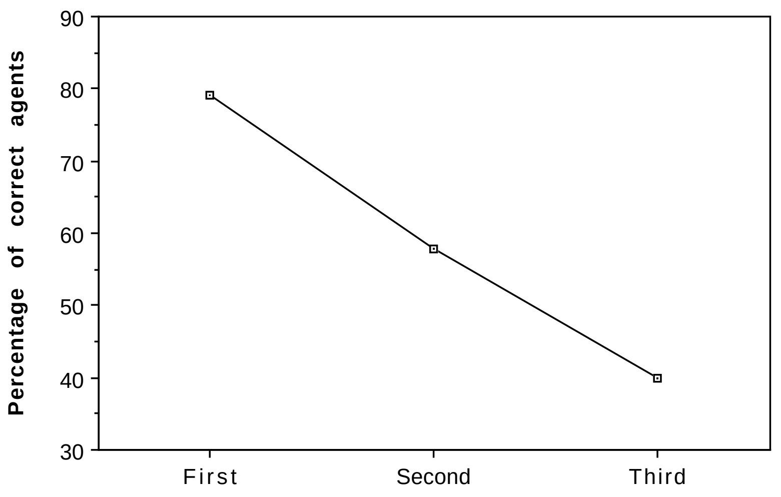

Figure 5: **Memory accuracy after return from center embeddings.** The percentage of correctly-remembered agents is plotted after the first, second, and the third pop in sentences with three levels of center embeddings. Each successive pop is harder and harder to do correctly. Similarly, SPEC remembers about 84% of the patients correctly after the first pop, and 67% after the second pop. The Stack representations were degraded with 30% noise to elicit the errors.

against noise. The average Parser error rose to 0.058, but the system still got 94% of its output words right, with very few errors in control. 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 5). 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. 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 sentence. It seems that traces for the earlier nouns are discernible in the previous-hidden-layer pattern, and consequently, they are favored at the output. Such priming effect is rather surprising, but it is very plausible in terms of human performance.

The semantic constraints (table 5) also have a marked effect on the performance. 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 (figure 6). 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`.

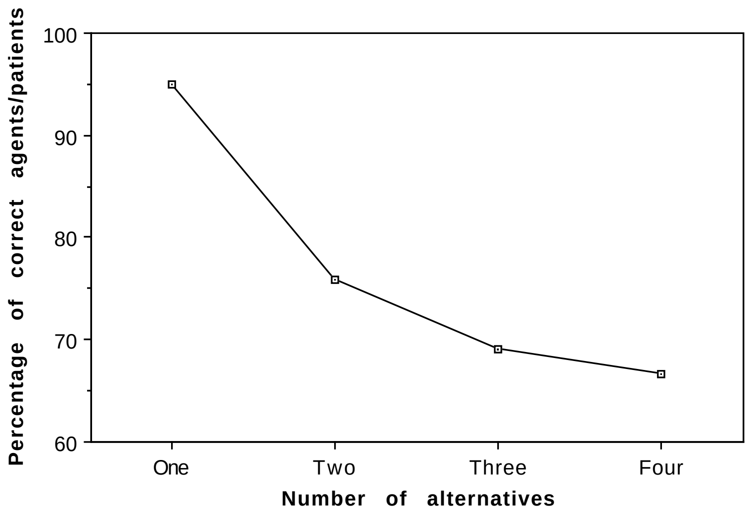

Figure 6: **Effect of the semantic restrictions on the memory accuracy.** The percentage of correctly-remembered agents and patients over the entire corpus is plotted against how strongly they were semantically associated with the verb. When there was only one alternative (such as `dog` as an agent for `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 degraded with 30% noise to elicit the errors.

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.

## 3.3 SPEC conclusions

The main insight from SPEC is that whereas soft constraints alone may not be sufficient for dynamic inferencing, when combined with a high-level control mechanism, novel inputs can be processed productively and systematically. Trained on only the basic sentence constructs, SPEC generalized to a wide range of new sentence structures. At the same time, the model maintained cognitively valid behavior that arises from combining soft constraints. Deep embeddings are difficult, and semantic constraints aid in interpreting the sentence. The errors that the model produces are similar to human performance. This way SPEC goes a long way in answering the challenge posed in the introduction: how seemingly symbolic behavior such as sentence processing can emerge from interacting subsymbolic soft constraints.

The Segmenter in SPEC is a first step toward high-level control in subsymbolic models. The Segmenter was designed specifically for this task: it monitors the input sequence and the state of the parsing network, and issues I/O control signals for the Stack memory and the Parser itself at appropriate times. It has a high-level view of the parsing process, and uses it to assign simpler tasks to the other modules. In this sense, the Segmenter implements a strategy for parsing sentences with relative clauses. It may be possible to build similar control networks for other linguistic tasks, or perhaps one sophisticated control network for language processing in general. Such control networks could play a major role in subsymbolic models of natural language processing and high-level reasoning in the future.

## 4 Future Outlook

If we adopt the view that language processing emerges from soft constraints, and that subsymbolic neural networks are a good model of this process, where will it take us? What are some of the future challenges and opportunities of this approach?

Although the subsymbolic models have been successful in demonstrating the utility of the approach, they are still mostly demonstrations of capabilities on toy problems. There are several technical challenges that need to be met before modeling language processing in a more realistic scale is possible. First, how can complex linguistic representations be encoded on subsymbolic neural networks? For example, how can an indefinite number of agents in a clause be represented, or clauses in a sentence, or sentences in a story, when there is a limited and fixed number of units in the network? Humans do not 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 generalize to new structures.

Second, 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. Building such corpora is very laborious, and it is unclear whether it is ever possible to have large enough training sets. It is also unclear what form the targets should take. In many cases, the targets represent some form of "language of thought," or internal representations of meaning, which are difficult to justify.

If these problems can be solved, however, it may be possible to build a subsymbolic model of sentence understanding which is capable of dynamic inferencing on general linguistic structure and which generates cognitively valid emergent behavior. The inferencing capabilities can be verified against the corpora. Cognitive validity can be tested at least in part by comparing the model with grammaticality judgements in humans (Gibson 1997). What is difficult for humans should also be difficult for the model.

The model could then be used to generate predictions on areas where not much is known about human performance and processes. For example, predicting semantic complexity of sentences, or why is `pot+dishwasher+police` harder than `diplomat+ballpark+princess`? Predictions could be made on how ambiguities are resolved in general, showing how soft constraints on syntax, semantics, frequency etc., interact. The model can be lesioned, the resulting behavior observed and matched with human impairments, and predictions on possible sources of deficits made. More efficient ways of retraining the model could translate to suggestions on better ways of rehabilitating aphasics. Such a model could serve as a platform for motivating and testing theories about human language processing and its breakdown.

## 5 Conclusion

This chapter has demonstrated, through subsymbolic neural network modeling, how sentence understanding could emerge from soft constraints. The constraints arise from regularities in the input, and are applied through correlating the current input with past contexts, as illustrated in the sentence disambiguation task. The same processes may be active in complex grammar processing as well, as demonstrated by the SPEC model of relative clauses. Augmented by a high-level control process, soft constraints combine to productive and systematic processing of new sentence structures, with plausible cognitive effects. This way a seemingly symbolic linguistic behavior may in fact be an emergent property of soft constraints. The approach may eventually lead to testable, cognitively accurate natural language processing models with good predictive power.

### Acknowledgements

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

## Note

Source code and on-line demos of the systems described in this paper are available in the World Wide Web under URL http://www.cs.utexas.edu/users/nn.

## References

Burgess, C., and Simpson, G. B. (1988). Neuropsychology of lexical ambiguity resolution. In Small, S. L., Cottrell, G. W., and Tanenhaus, M. K., editors, *Lexical Ambiguity Resolution: Perspectives from Psycholinguistics, Neuropsychology & Artificial Intelligence*, 411–430. San Mateo, CA: Morgan Kaufmann.

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.

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

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

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

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.

Forster, K. I., and Bednall, E. S. (1976). Terminating and exhaustive search in lexical access. *Memory and Cognition*, 4:53–61.

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.

Gibson, E. (1997). Syntactic complexity: Locality of syntactic dependencies. Manuscript.

Glucksberg, S., Kreutz, R. J., and Rho, S. (1986). Context can constrain lexical access: Implications for interactive models of language comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 12:323–335.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation.* Reading, MA: Addison-Wesley.

Hinton, G. E., and Shallice, T. (1991). Lesioning an attractor network: Investigations of acquired dyslexia. *Psychological Review*, 98:74–95.

Hogaboam, T. W., and Perfetti, C. A. (1975). Lexical ambiguity and sentence comprehension. *Journal of Verbal Learning and Verbal Behavior*, 14:265–274.

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.

Lange, T. E., and Dyer, M. G. (1989). High-level inferencing in a connectionist network. *Connection Science*, 1:181–217.

MacWhinney, B. (1977). Starting points. *Language*, 53:152–168.

MacWhinney, B. (1988). The processing of restrictive relative clauses in Hungarian. *Cognition*, 29:95–141.

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. (1990). A PDP architecture for processing sentences with relative clauses. In Karlgren, H., editor, *Proceedings of the 13th International Conference on Computational Linguistics*, 201–206. Helsinki, Finland: Yliopistopaino.

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.

Onifer, W., and Swinney, D. A. (1981). Accessing lexical ambiguities during sentence comprehension: Effects of frequency of meaning and contextual bias. *Memory and Cognition*, 9:225–226.

Plaut, D. C. (1991). *Connectionist Neuropsychology: The Breakdown and Recovery of Behavior in Lesioned Attractor Networks.* PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. Technical Report CMU-CS-91-185.

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

Schvaneveldt, R. W., Meyer, D. E., and Becker, C. A. (1976). Lexical ambiguity, semantic context, and visual word recognition. *Child Development*, 48:612–616.

Seidenberg, M. S., Tanenhaus, M. K., Leiman, J. M., and Bienkowski, M. (1982). Automatic access of the meanings of ambiguous words in context: Some limitations of knowledge-based processing. *Cognitive Psychology*, 14:489–537.

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.

Simpson, G. B., and Burgess, C. (1985). Activation and selection processes in the recognition of ambiguous words. *Journal of Experimental Psychology: Human Perception and Performance*, 11:28–39.

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.

Tanenhaus, M. K., Leiman, J. M., and Seidenberg, M. S. (1979). Evidence for multiple stages in the processing of ambiguous words in syntactic contexts. *Journal of Verbal Learning and Verbal Behavior*, 18:427–440.