Connectionist Natural Language Processing

Erik Hjelmas*
Department of Electrical Engineering and Science
Gjovik College
P. O. Box 191, N-2800 Gjovik
NORWAY

ABSTRACT

This paper focuses on connectionist models in natural language processing. We briefly present and discuss several aspects of high level tasks which recently have been approached with connectionism, either with localist or parallel distributed processing models. Several interesting architectures have been proposed in the last decade, and connectionist natural language processing seems like a promising area for future research in artificial intelligence.

1. Introduction

The field of Natural Language Processing (NLP) is being approached in several different ways. Most NLP systems so far have been implemented with a symbolic approach, which is the more "traditional" approach to Artificial Intelligence (AI). After the revival of connectionism in the 1980s, when Rumelhart & McClelland[20] demonstrated that it was possible to get around the limitations which Minsky & Papert[15] pointed out in the 1960s, papers on connectionist approaches to NLP started to appear.

Currently, it is well known that connectionist models perform very well at low-level tasks, such as speech recognition and speech generation. But these areas are not what we consider as NLP in this paper (even though they would be considered as an important part in many large-scale NLP systems). Our approach to NLP in this paper is based on Allen[1].

What makes NLP worth doing with connectionism? The motivation behind this approach is that there are several issues that the symbolic approach does not address. Symbolic systems cannot easily utilize statistical properties of the data to improve processing. Symbolic models are not capable of explaining certain aspects of human performance because they are high-level models, not very similar to the physical structure that implements the processes in the brain. Explaining issues like the origin of performance errors, or what happens when the system is corrupted with noise, is therefor problematic. Connectionist models, on the other hand, are very similar to the physical structure in the brain, and are capable of giving plausible explanations to several issues where symbolic models cannot.

* email: erikh@hig.no

2. Low-Level Tasks

Low-level tasks in NLP are problems that are limited to a domain that does not need much context, and are typically feasible as a small part of a larger system. These tasks can be problems like syntactic parsing, simple ambiguity resolution in a specified context, limited domain translation tasks, etc[1].

Since low-level tasks normally have very little internal structure and many times can be reduced to "simple" mappings, they can be solved in a feasible approach with connectionism. Rumelhart & McClelland[20] gave a good example of this when they trained a neural network to learn the past tense forms of English verbs. Their model was able to identify the correct past tense of verbs it had not been exposed to during training, making its behavior similar to humans learning language.

Connectionist models capable of syntactic parsing, together with a limited capability of semantic interpretation and disambiguation, have appeared in several papers[3, 21]. The major accomplishments with these models is their human-like performance. The natural ability to generalize in connectionism makes

[1] Actually, the term low-level tasks are generally used to refer to speech processing and similar problems, while we in this paper use it to refer to the "mid-level" tasks of NLP.

connectionist models biologically plausible, and able to explain human performance at certain tasks. Hanson & Kegl[7] proposed a connectionist network capable of learning natural language grammar from exposure to natural language sentences. Their model was able to recognize sentences that it had either seen before, or that it might have seen before, making it similar to the performance of an associative memory where a large number of patterns have been stored.

Problems with parsing and semantic interpretation with connectionist models arise when the system is required to handle deviations in the input (the presented stimuli). For example, semantically reversible sentences such as "John loves Mary" makes distinguishing Agent from Object a problem. To handle such deviations, the model has to have some internal structure, which makes the connectionist architecture complex.

3. High-Level Tasks

There is clearly advantages in the connectionist approach at a low level of natural language processing, but can high-level tasks also be modeled in connectionism? The fundamental problem underlying this question is that high-level cognitive tasks are often composed of subtasks. In NLP these would be tasks such as parsing, memory storage and retrieval, reasoning from the discourse, and generating language. The complex behavior of a composite task cannot be done in a simple mapping, it calls for a structured architecture. This is not intuitively feasible in connectionism, but might still be possible.

The two major architectures for connectionist NLP is localist and distributed models. The classical PDP models are distributed and are typically referred to as neural networks. Localist models are based on semantic networks similar to symbolic networks, but do not have labeles arcs, and are characterized by spreading activation and constraint satisfaction methods for retrieval.

3.1. Variable Binding

One of the great powers of symbolic processing is binding variables to values real-time[4]. Values, from simple datatypes to complex structures, can easily be assigned to variables through the use of rules. In NLP, symbolic rules typically establish relations between agents and objects, or bind values such as names to agents. This is not a very natural capacity of connectionist models, but several mechanisms have been proposed to deal with these issues.

The nodes in localist networks represents entities from NLP such as predicates and roles. Without any mechanism for variable binding, these networks would have to represent all possible combinations of relations for the spreading activation to work appropriately with respect to the desired relationships in the system. One solution to this problem is the use of signatures[25]. With signatures all the nodes in the network are assigned an identifying activation value (a signature). Rules are represented by pathways (separate from those used for spreading activation) between nodes that represent predicates. If signatures are propagated between these pathways at the same time as spreading activation occurs, this can be interpreted as the firing of the rule represented by that pathway. The relationship resulting from that rule, is identified by the signatures arriving from the nodes involved. E.g., if the role BUYS:BUYER propagates its signature (e.g. 18) to OWNS:OWNER and BUYS:OBJECT propagates its signature (e.g. 11) to OWNS:OBJECT, during spreading activation, then the network can be interpreted as firing a rule "John" (BUYER) buying a "TV" (OBJECT) results in "John" owning a "TV".

Phase synchronization is a different approach to this problem of dynamic variable binding[22]. A phase is a subunit of the time it takes for activation to spread between units. The number of phases for one spread of activation limits the number of bindings that can be propagated through the network. Each node that has to be bound is assigned one phase. The nodes that receive activation operates such that the phases are delayed exactly one step of spreading activation, in this way synchronizing the phases throughout the network. The variable binding in phase synchronization occurs when instances are assigned to phases. E.g. "John" might be assigned phase 2, and "TV" assigned phase 5. If the total number of phases are 8, this means that the network can only handle 6 more bindings. This might be enough for retrieval tasks, but when dealing with more complete NLP, problems will arise with sentences like:

> The large, skinny man gave a small, red firetruck to the eight-year old son of his friend Jim's cousin with the blond hair.

The major difference between these two approaches is that an unbounded number of signatures can be propagated through the network, while a very limited number of phases is possible. Thus, phases works well for small retrieval tasks, but cannot deal

with more complex language processing. However, the biggest problem both these approaches face is that arbitrarily complex structures cannot be bound to variables without dynamically creating new nodes in the network, and this would violate the connectionist paradigm from neuroscience.

In distributed connectionist networks, this problem should be easier to solve since dynamic modification of the network is more biologically plausible when the representation is distributed (vector representation). Two major approaches have been proposed for variable binding in these networks.

Tensor products[23] are products of vectors leading to a new vector of higher rank (dimensionality). Tensors are created to represent bindings between distributed representations. E.g. if a n-dimensional vector V (representing a role), and a m-dimensional vector W (representing its filler) are bound together, a tensor (in this case a matrix) is created from the outer product VW.

A different idea is to use $ID + ContentVectors$. The representation vector is split, such that one part represents the general object, and the other part represents the specific instance. By using this approach in a modular architecture, the system can propagate bindings which gives it the capacity to make inferences[14]. This approach can be implemented by training PDP networks to learn the identity mapping of each vector, in other words the network can map the entire vector without altering the ID part. This means that the content can change in a mapping while the identity stays the same.

Both these approaches provide the capability of dynamically creating bindings, but their performance is still very limited. By using tensor products, the system becomes very symbolic in its behavior (more sequential than parallel processing), which causes it to loose some of its nice distributed features (generalization). The $ID + Content$ approach is also nice, and works well for limited domains, but the question is if it can scale up when more complex and strict variable bindings are required.

3.2. Scripts

Simple stereotypical narratives such as

> Jim went to Leone's. Jim asked the waiter for steak. Jim left a small tip.

imply much more than they actually state, but all in a limited domain (in this case; eating at a restaurant). The underlying structure representing the complete domain is called a script. Miikkulainen[12] developed a NLP system called DISCERN (DIstributed SCript processing and Episodic memoRy Network) capable of answering questions and making inferences from scripts. DISCERN was based on "symbolic thinking" in its overall architecture, but built entirely from connectionist modules. In DISCERN, the entire high-level task is implemented with PDP mechanisms. The system processes scripts by establishing causal chains of the scripts, e. g. one combination of customer, restaurant and food instantiates one specific story. DISCERN was trained with three different scripts, each with three tracks: restaurant (fancy, coffee-shop, and fast-food), shopping (clothing, electronics, and grocery tracks), and travel (plane, train, bus tracks).

DISCERN consists of four subsystems: parsing, generating, question answering, and memory. Each of these subsystems are divided into two modules for input and output. The modules are trained in their tasks both in parallel and separately. Pattern transformation in DISCERN are carried out in the parsing, generating, and question answering modules. These modules are feed-forward neural networks or simple recurrent networks[5], and are trained with backpropagation. The memory modules are implemented with feature maps[9], and are responsible for storing and retrieving patterns.

When an input-sequence (of sentences) is presented to DISCERN, the sentence-parser processes it word-by-word and forms case-role representations. These representations are then sequentially processed by the story-parser, which forms an assembly-based representation of the whole story as its output. This can be viewed as a data-specific slot-filler representation[12], with respect to script, track and role-bindings. It constitutes the internal representation of the story, and is stored at a location in episodic memory determined by self-organizing feature maps. When DISCERN is asked a question, the question is read word-by-word, and the sentence-parser forms a cue to episodic memory. Episodic memory then retrieves its best guess, and feeds this to the story generator and sentence generator, which produces the appropriate answer in the answer-producer-module. DISCERN is also capable of generating "the complete story", after being presented with a few input sentences (the following example is from [12]):
Input:

```
JOHN WENT TO MAMAISON.
JOHN ASKED THE WAITER FOR LOBSTER.
JOHN LEFT A BIG TIP.
```

Output from DISCERN's story generator:

```
JOHN WENT TO MAMAISON.
THE WAITER SEATED JOHN.
JOHN ASKED THE WAITER FOR LOBSTER.
JOHN ATE A GOOD LOBSTER.
JOHN PAID THE WAITER.
JOHN LEFT A BIG TIP.
JOHN LEFT MAMAISON.
```

This system is probably the highest level connectionist system built so far for NLP. It performs very well, and by using this architecture DISCERN shows

- that script-based inferences can be learned from experience
- how episodic memory can automatically be organized
- how word semantics can be learned from examples of their use

Scripts are considered as NLP tasks at a high level, so DISCERN is a good example of the capacity of connectionism. But DISCERN is also a good example of the limitations of connectionism because, as indicated earlier, deviations in the input create problems for these models. This is also the case for DISCERN.

3.3. Representing Structure

There is some question about whether scripts really are a representable task for high-level NLP. The reason for this is that a high-level NLP task should require a dynamic data structure which is not the case for scripts (D. Metzler, personal communication). Scripts only require an almost static structure in its data representation.

So what kind of NLP-related structure can connectionist models represent? Elman[5] gives a very good example of how time, in terms of sequential events, can be represented using a recurrent neural network architecture. He shows how a simple recurrent network (SRN) is capable of learning to predict words in sentences. A SRN is a feed-forward neural network which has been augmented with a extra set of connections in the hidden layer feeding back to the same hidden units. The idea is that the hidden units have to include the output of the previous processed pattern when processing the current pattern. This creates a link between patterns in time (sequentially). Cleeremans et al. [2] continued along the same lines as Elman, and presented neural networks capable of learning a finite state grammar by using SRNs.

The problem is that when scaling up to larger NLP systems, the data structure becomes extremely complex. Language has so many deviations and statistical irregularities that an overall architecture based solely on connectionism will be infeasible with the existing connectionist models presented so far.

3.3.1. Recursive Structure

Recursive structure is a typical example of a high level task in NLP. Sentences with embedded recursive structures like:

> Jim told Sara that Pete told Linda that Jenny told John that Frank ran away.

can even be confusing for humans if the depth of recursion in the sentence is substantial[2].

Localist networks can propagate structures by the use of signatures or phases, but this causes problems when the structures become recursive. In localist networks, predicates can be represented with separate nodes, and recursion can in the same manner be obtained by using multiple copies of nodes, but this is not a "natural" solution in connectionist networks, as mentioned earlier.

As previously mentioned, Elman[5] came up with a neural network architecture for incorporating time in the model. The same network can also be used for representing recursive structure, because the architecture allows the network to feed back to itself. Another approach, called Recursive Autoassociative Memory (RAAM), developed by Pollack[18], is perhaps even more suitable for representing recursive structure. RAAM is an encoder network, which means that it maps sentences (or clauses) to themselves, in other words, the stimuli is equivalent to the target. The characteristics of the RAAM is that the hidden layer representation (which is a compressed version of the input sentence) can be fed into the network together with the next presented input (similar thinking as in SRNs). This means that there is a connection between the clauses allowing the representation of recursive structure. Miikkulainen[13] made use of RAAM as a stack in a distributed connectionist network architecture called SPEC. The SPEC system succesfully parses sentences with relative embedded clauses, and shows that learning and applying grammatical structure for parsing is possible with only distributed networks. SPEC performed without mistakes on a corpus of 98100 sentences, and the system did not only generalize to new sentences of famil-

[2] Normally a depth of three is enough to confuse the reader.

iar templates, but also to new templates. However, SPEC's domain was limited to sentences with relative embedded clauses, so fundamental issues in connectionist NLP, such as processing exceptions (deviations) and representing flexible structure was not addressed in this system.

Both these architectures show promise for dealing with recursive structure. RAAM has probably been the most implemented of the two. With the way RAAM works, it can function as a stack and encode lists and trees, and if long-term memory is added, it also has the capacity to store graphs. Since RAAM does not have any particular limits set with respect to size of stimuli or depth of recursion, it should be able to scale up to deal with large NLP tasks (rather than just limited domains such as scripts).

3.3.2. Lexical Memories

Lexical memory is an essential part of the NLP system. Traditionally, lexical memory has been implemented symbolically in a frame-like structure with attached rules. In this approach words have a static internal representation, so this approach can also work when implemented in a localist connectionist network since words can be represented by nodes in the same static way similar to symbolic systems. In PDP networks lexical entries are formed from examples (activation vectors), which has the pleasant effect that words with similar meanings end up with similar vector representations. Miikkulainen's[12] FGREP method and Lee's[11] xRAAM method have been developed for automatically forming lexical entries with the PDP approach.

The FGREP method can be implemented as modules (separate PDP networks) working on a global lexicon (DISCERN[12], DISPAR[14]). FGREP is a feed-forward neural network which uses an extended version of the backpropagation learning algorithm. It is extended in the sense that the error signal which backpropagation is based on is propagated all the way back to the input layer, so the actual input pattern is also modified together with the network[3]. The neural network does not have to be a strict feed-forward network. FGREP modules can operate on sequences of words if the neural network is recurrent (e.g. SRN[5]). Tasks like case/role assignment and mapping an event to a sequence of words are typical examples where FGREP modules work well.

Lee's[11] xRAAM is similar to RAAM (Recursive AutoAssociatve Memory, described earlier), the only difference is that xRAAM is augmented with lexical memory. The patterns to be stored in lexical memory are the hidden layer activations for the words when the network converges for a training set. A training set for the word *soda* might look like the following:

```
SODA IS A DRINK
SODA HAS BUBBLES
SODA IS CONTAINED IN BOTTLES AND CANS
```

If one of the words is not encoded in the lexicon yet (e.g. *drink*), *soda* has to be retrained (modified) again after that word has completed the same encoding process. xRAAM assumes that the final hidden layer activation pattern obtained for a word is the representation appropriate for that word in the lexicon.

Both FGREP and xRAAM developes distributed representation patterns for the lexical entries. This approach works well because the representations can be passed to different modules in the NLP system, and they encode the propositional contents of the words. Since the patterns are distributed, they also possess the other nice features of patterns from PDP networks (generalized patterns, noise-resistant, etc.).

3.3.3. Semantic and Episodic Memories

For a NLP system to be called a good AI system, it has to contain several components which are necessary in an intelligent system. Two important components of an intelligent system are semantic and episodic memory. When processing language there is both a need for forming semantic knowledge and to store specific episodes in memory. Localist connectionist networks can easily form semantic memory, since the nature of the model is similar to a semantic network. Creating episodic memory, on the other hand, is more problematic. This requires specific instances of episodes which would require the creation of new nodes in the localist network, and this violates the connectionist paradigm (as described earlier).

PDP networks can also handle word-semantics "easily", since this is one of the nice features of these models. Semantic memory is represented by the weights in the network, which are modified through training. PDP models have, just like localist models, problems with handling episodic memory. There have been one implementation of episodic memory in connectionist NLP, namely DISCERN[12]. DISCERN uses hierarchical feature maps to organize episodic memory. This works well in this system, but again

[3] Normal backpropagation only propagates the error signal back to the first hidden layer in the network

the problem is if it will scale up to larger NLP systems. DISCERN works only with scripts, and the instances of episodic memory it handles are similar in structure and content, which may not be the case in other NLP systems.

When considering this question of episodic memory, we encounter one of the key differences between symbolic and connectionist systems. Episodic memory requires specific instances, normally with sharp boundaries of the knowledge (not always necessary, but there are not a fluent connection between different episodes). Symbolic systems are powerful, and can handle this easily in a frame-like structure. There is no problem with instantiating new episodes and incorporating them into memory. Connectionist models cannot create new nodes "on the fly", they have to modify a distributed representation or the weights of a network (in PDP models), which will affect the rest of the system, since there are no strict boundaries for the represented knowledge. This is one aspect of AI where symbolic systems seem more plausible as a model of human cognition than connectionism, since humans have very distinct memory of specific episodes.

3.4. Control

It is obvious that a large NLP system cannot be built from a single neural network performing just one mapping, it has to be more complex, and the best approach (from a connectionist point of view) seems to be separate modules performing different tasks connected together in a specified architecture. This raises the question of how to build this architecture, and how to control the dataflow between the modules. One solution is to use a localist connectionist network with millions of nodes, each representing different instances or types from language. In this solution, each node would represent a module in the network. However, this raises several problems like combinatorial explosion of nodes (thus modules) and dynamically creating new nodes.

A different approach from the systems described earlier in this paper is taken in the DCAIN system[24]. DCAIN is a distributed connectionist network with potentially many hundreds of ensambles of units, which are organized in a form similar to semantic networks, thus are labeled Parallel Distributed Semantic (PDS) networks. Each ensamble of units is connected to other ensambles through propagation filters. These propagation filters control the routing of data in the system, and are themselves learnable ensambles of units. When an activation vector arrives at a propagation filter, the vector is routed along the network if every unit in the ensamble is driven above its threshold.

The architecture used in DCAIN solves the problem from localist network with a combinatorial explosion of nodes, because specific instances are represented by different activation vectors, so dynamically creating new nodes is not necessary. E. g. different instances of $xBUYSy \rightarrow xOWNSy$ is represented by different patterns in the BUYS ensamble. Another advantage with this approach is that the structure of this architecture corresponds better to how the brains appears to be organized, compared to localist networks and simple PDP networks.

Learning is also faster in PDS networks than in PDP networks because the architecture allows for parallel training of ensambles. Since PDS networks is based on distributed connectionist networks for all subtasks, they possess most of the advantages of these models (generalization, noise-tolerance, etc).

3.5. Perception

Perception related to NLP is also a high level task, even though it may not be the first thing that comes to mind, it is known that children learn language semantics in part by associating verbal sentences with visual stimuli. Children who have learned nouns like *ball* and *boy* can begin to learn the meaning of *kick*, by observing a boy kicking a ball when they hear the sentence *the boy kicks the ball*.

One relatively simple system developed to simulate this phenomena was developed by Regier[19]. Regier developed a connectionist network which learned the meaning of phrases by associating them with a visual stimuli consisting of two objects (one stationary, one moving) in a 2-D microworld. This system is a part of the research in the L_0 project[6] which has the long-term goal of acquiring language via association with perception. A larger system called DETE has been developed by Nenov & Dyer[16, 17]. DETE can also learn the meaning of sentences (or sequences of words, clauses) via association with visual stimuli. The visual stimuli presented to DETE can have up to five different objects ("blobs"). During training, DETE recieves verbal stimuli describing the visual stimuli. DETE is capable of performing two main tasks: (a) Verbal-to-visual association when given a verbal stimuli, and (b) Visual-to-verbal association when given a visual sequence.

The tasks performed by DETE demonstrates an important relationship between visual and verbal stimuli, but it is only implemented on relatively low-

level language. When a system like this is scaled up to a higher level language domains it will also encounter the problems of representing structure indicated earlier in this paper. However, if the work along the lines indicated in section 3.3 and 3.4 will lead to systems capable of handling large scale applications, the performance of systems like DETE would be even more impressive.

4. Hybrid Approach

Language is both symbolic and subsymbolic. It is symbolic because, obviously, language consists of symbols. It is also subsymbolic because of the ambiguity behind the symbols and the fuzziness of concepts. Kwasny & Faisal[10] proposed a hybrid approach to natural language parsing. In their architecture, a neural network was used to parse the input sentence, while the parsed symbols are handled with a symbolic buffer and a stack. This approach is gaining popularity and also seems like a very feasible approch to cognitive modeling since the low-level task (connectionist model) is "similar" to the physical structure of the human brain, while the high-level task is symbolic which seems to be the most plausible explanation to high-level human cognition[12].

Hybrid models can exploit the advantages of several different approaches in the same system. NLP is such a complex task that it might be a hybrid model that exhibits the best performance. PDP networks are good at learning, capturing statistical regularities in data, robust processing, associative retrieval, data fusion and generalization. Localist connectionist networks are suitable for parallel constraint satisfaction, and symbolic systems can represent and apply rules, schemas, variable binding and sequential control naturally. If a system can be built that captures all these strengths, it might be able to scale up as a system capable of handling large-scale NLP.

Another advantage with using connectionist models as a part of a NLP system, is the potential for rule extraction. If the architecture of the system is symbolic, using an neural network to extract grammar rules might still be a good idea. By pruning the neural network and examining the hidden unit space activation (for instance with cluster analysis), useful information about the input sentences can be revealed[8]. Different languages have different degrees of complexity and in certain cases this could be a very useful approach.

5. Conclusion

We have presented a brief overview of connectionism in natural language processing and discussed several areas of high-level NLP where connectionist approaches have been proposed. It seems like most NLP tasks can in some way be solved in connectionism, but the proposed solutions is not always intuitively easy to scale up and fit into a composite system. Because of the limited capability of connectionist models the represent complex structure, high-level NLP tasks seems to be best handled with a symbolic approach. But with symbolic "thinking" in an architecture built from connectionist modules, a complete, integrated NLP system can be implemented in the connectionist approach, and the fact remains that connectionism has so many nice features in several aspects of processing that future research could lead to better NLP systems.

References

[1] J. Allen. *Natural Language Understanding.* Benjamin/Cummings, Redwood City, CA, 1995.

[2] A. Cleeremans, D. Servan-Schreiber, and J. L. McClelland. Graded state machines: The representations of temporal contingencies in feedback networks. In Y. Chauvin and D. E. Rumelhart, editors, *Backpropagation: Theory, Architectures and Applications.* Hillsdale, NJ: Lawrence Erlbaum Associates, 1995.

[3] G. W. Cottrell. Connectionist parsing. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum, 1985.

[4] M. G. Dyer. Connectionist natural language processing: a status report. In R. Sun and L. A. Bookman, editors, *Computational Architectures Intergrating Neural and Symbolic Processes.* Norwell, MA: Kluwer Academic Publishers, 1995.

[5] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[6] J. A. Feldman, G. Lakoff, A. Stolcke, and S. Hollback Weber. Miniature language acquisition: A touchstone for cognitive science. Technical report, ICSI, Berkeley, CA, 1990.

[7] S. Hanson and J. Kegl. Parsnip: A connectionist network that learns natural language grammar from exposure to natural language sentences. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum, 1987.

[8] C. L. Harris. Connectionism and cognitive linguistics. *Connection Science*, 2:7–33, 1990.

[9] T. Kohonen. *Self-Organization and Associative Memory*. Springer Verlag (2nd ed.), 1988.

[10] S. C. Kwasny and K. A. Faisal. Connectionism and determinism in a syntactic parser. *Connection Science*, 2:63–82, 1990.

[11] G. Lee, M. Flowers, and M. G. Dyer. Learning distributed representations for conceptual knowledge and their application to script-based story processing. In N. Sharkey, editor, *Connectionist Natural Language Processing: Readings from Connection Science*. Norwell, MA: Kluwer Academic Publishers, 1992.

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

[13] R. Miikkulainen. Subsymbolic parsing of embedded structure. In R. Sun and L. A. Bookman, editors, *Computational Architectures Intergrating Neural and Symbolic Processes*. Norwell, MA: Kluwer Academic Publishers, 1995.

[14] R. Miikkulainen and M. G. Dyer. Natural language processing with modular pdp networks and distributed networks. *Cognitive Science*, 15:343–399, 1991.

[15] M. L. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

[16] V. I. Nenov and M. G. Dyer. Perceptually grounded language learning: Part 1 - a neural network architecture for robust sequence association. *Connection Science*, 5, 1993.

[17] V. I. Nenov and M. G. Dyer. Perceptually grounded language learning: Part 2 - dete: A neural/procedural model. *Connection Science*, 5, 1993.

[18] J. B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 46:77–105, 1990.

[19] T. Regier. *The Acquisition of Lexical Semantics for Spatial Terms: A Connectionist Model of Perceptual Categorization*. PhD thesis, University of California at Berkeley, 1992.

[20] D. E. Rumelhart and J. L. McClelland. On learning the past tenses of english verbs. In J. L. McClelland and D. E. Rumelhart, editors, *Parallel Distributed Processing, Vol II: Psychological and Biological Models*. Cambrigde, MA: MIT Press, 1986.

[21] B. Selman and G. Hirst. A rule-based connectionist parsing system. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum, 1985.

[22] L. Shastri and V. Ajjanagadde. From simple associations to systematic reasoning. a connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavoral and Brain Sciences*, 16:417–494, 1993.

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

[24] R. A. Sumida and M. G. Dyer. Propagation filters in pds networks for sequencing and ambiguity resolution. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*. San Mateo, CA: Morgan Kaufmann Publ., 1992.

[25] R. Sun. On variable binding in connectionist networks. *Connection Science*, 4:92–124, 1992.