A Connectionist Architecture for Sequential Symbolic Domains

Ajay N. Jain

December, 1989
CMU-CS-89-187$_2$

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

This report describes a connectionist architecture specifically intended for use in sequential domains requiring symbol manipulation. The architecture is based on a network formalism which differs from other connectionist networks developed for use in temporal/sequential domains. Units in this formalism are updated synchronously and retain partial activation between updates. They produce two output values: the standard sigmoidal function of the activity and its velocity. Activation flowing along connections can be gated by units. Well-behaved symbol buffers which learn complex assignment behavior can be constructed using gates. Full recurrence is supported. The network architecture, its underlying formalism, and its performance on an incremental parsing task requiring non-trivial dynamic behavior are presented.

This research was funded by grants from ATR Interpreting Telephony Research Laboratories and the National Science Foundation under grant number EET-8716324. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of ATR, NSF, or the U.S. Government.

Contents

List of Figures

1 Introduction

There has been much recent work developing connectionist formalisms which can be applied to sequential domains. These approaches range from recurrent extensions of back-propagation [5,12] to networks in which time is represented spatially (for examples, see [4,6,8]). In this report, a network architecture based on a new network formalism is described. It has been specifically designed for handling sequential symbolic input. The architecture has been applied to incremental parsing of sentences.[1]

Traditional methods employed in parsing natural language have focused on developing powerful formalisms to represent syntactic and semantic structure along with rules for transforming language into these formalisms. The builders of such systems must accurately anticipate and model all of the language constructs that their systems will encounter. Spoken language, with its weak grammatical structure, complicates matters. Connectionist networks which learn to transform

input word sequences into meaningful target representations may be useful in such domains.

Application of connectionist computational models to language tasks is not a new idea. Some researchers have used connectionist networks to implement formal grammar systems for use in syntactic parsing [1,7,17,9]. These networks do not learn their grammars. Other work has focused on semantics but either avoided the parsing issue, or the networks did not *learn* to parse [13,18,2,3]. Stochastic modeling of language is another approach to spoken language tasks. Using hidden Markov models, it is possible to statistically model word sequences with direct application to speech recognition [11]. Connectionist networks have also been applied in building statistical language models for use in speech recognition [15]. However, such language models do not produce traditional parses of sentences; they seek to use statistical regularities in language structure and word usage directly for recognition tasks.

The network architecture presented here learns its own "grammar rules" through training on example sentences. The trained networks transform an input sequence of words into a syntactic/semantic target representation. This work is a small first step in learning to build connectionist networks for parsing which will be useful in realistic natural language domains.

2 Incremental Parsing

Language is inherently sequential. Interpreting a sequence of words requires the incorporation of right context information into an evolving representation of meaning. Connectionist networks have been applied to many types of static recognition problems, but problems such as language interpretation require complex dynamic behavior over time. The following sentences illustrate the point:

- A snake ate the girl.
- A snake was given to the dog by Peter.

In the first sentence, "a snake" acts as an agent, but in the second sentence, "a snake" is a patient. It is not possible to determine the proper role of a noun phrase based on left context information alone—right context is needed. Processing sentences whole instead of one word at a time alleviates this problem but is undesirable especially in spoken language processing.

Below, a parsing task is presented. It is designed to see if connectionist networks can learn the dynamical behavior necessary to process sentences incrementally. Following that is a brief overview of the connectionist architecture used along with its temporal performance on the first sentence given above. The remaining sections of the report describe the details of the network formalism and architecture and its performance on the task.

[1]The parsing architecture was first presented in [10].

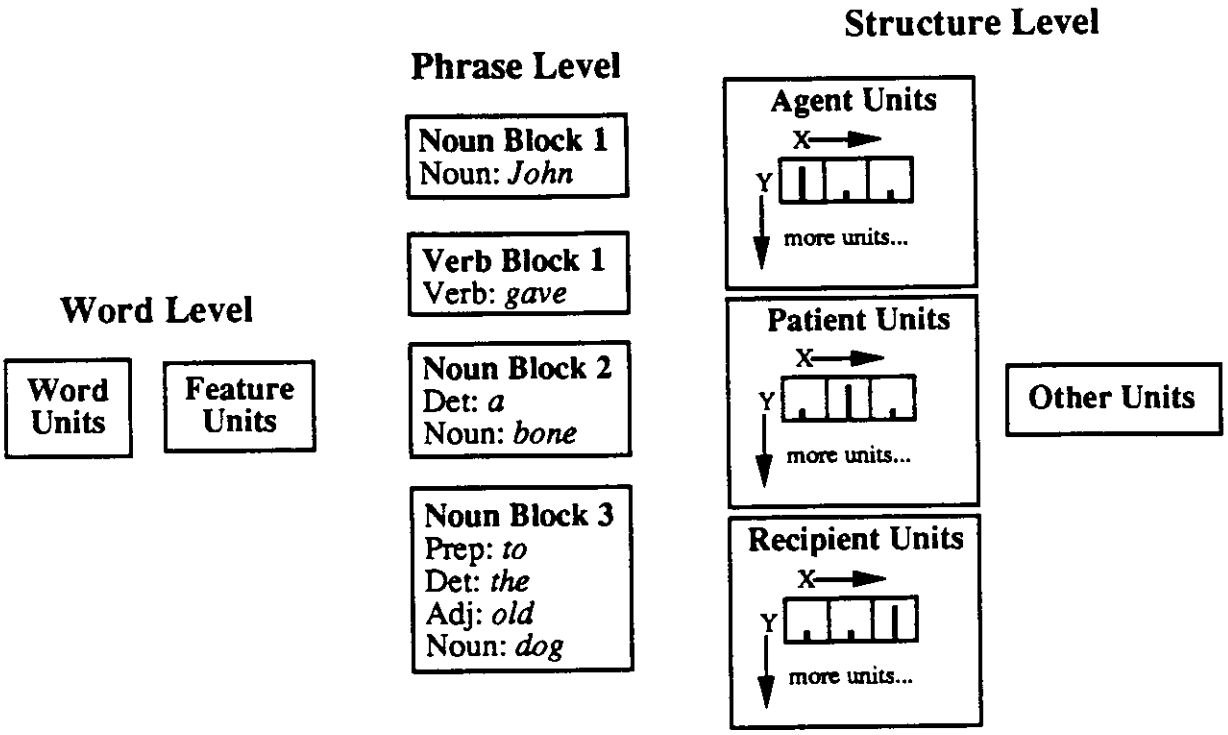

Figure 1: High-Level Network Architecture.

2.1 Task Description

The domain for this experiment consists of active and passive sentences consisting of up to 3 noun phrases and 2 verb phrases each. There are up to three roles for nouns to fill for each verb—agent, patient, and recipient. The lexicon consists of 40 words. The emphasis of the experiment is on single clause sentences. Two clause sentences are present to suggest extensibility.

The input for the network is sequential. Words are presented to the network one at a time. The network must learn to parse the input stream into noun and verb blocks consisting of head words and their modifiers. Further, noun phrases must be assigned an appropriate role or attached to another noun phrase. Subordinate and relative clauses and their relationships must be identified.

2.2 Overview of Network Architecture

Figure 1 gives a high-level picture of the network architecture along with the target representation for "John gave a bone to the old dog." There are three levels to the network: Word, Phrase, and Structure. The Word level contains the representation for the current input word. A word is presented to the network by stimulating its associated word unit for a short period. This evokes a pattern of activation across the feature units which represents the meaning of the word. The Phrase level learns to use the time-varying word representation to build noun and verb blocks which represent the phrasal structure of the input sentence. The Structure level learns to assign roles to noun phrases and identify phrasal/clausal relationships.

The sentence shown in Figure 1 is broken up into 4 blocks in the Phrase level: "[NB1 John] [VB1 gave] [NB2 a bone] [NB3 to the old dog]." The groups of units in the Structure level show the role assignment for this simple single clause sentence. A high activation level of a unit X,Y in the group of units marked Agent indicates that the network believes that Noun Block X is the agent of Verb Block Y.

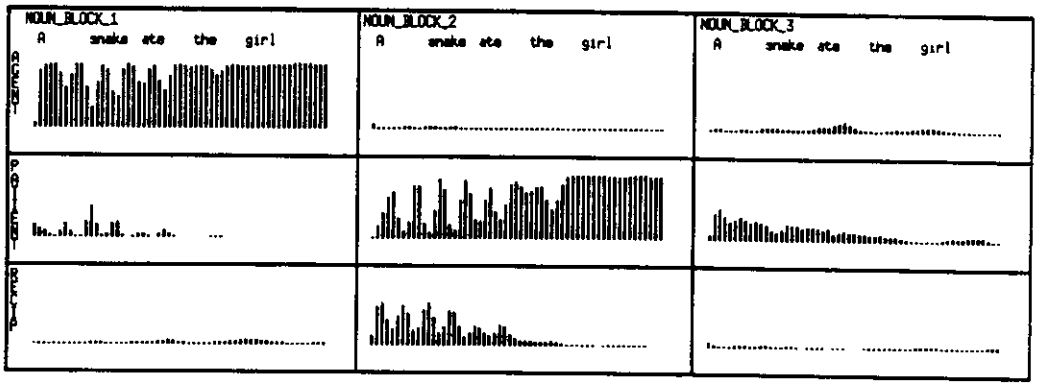

Figure 2: Performance of the network on, "A snake ate the girl."

The units in the groups marked Patient and Recipient have analogous meanings. In the diagram, "John" is labeled as the agent, "a bone" the patient, and "to the old dog" the recipient. The other groups of units will be discussed later.

Recall the first example sentence given at the beginning of this section, "A snake ate the girl." It is a simple active sentence with an Agent/Patient role structure. For this sentence, the behavior of the Structure units corresponding to the roles of Verb Block 1 in a trained network are shown in Figure 2. Each small box of vertical lines corresponds to the activation level of a single unit through time (long lines for high activity, time proceeds from left to right). The unit activation pattern in the upper left box indicates the hypothesis that Noun Block 1 functions as the Agent in the sentence. Activation patterns which form columns correspond to the label at the top of the uppermost box. Rows of activation patterns correspond to the label in the leftmost box. There are three columns—corresponding to the three noun blocks. There are three rows—corresponding to the the role values.

During presentation of the first word, the unit representing the hypothesis that Noun Block 1 is the Agent (row 1, column 1) becomes quite active. The network "expects" that the sentence is active (due to a high proportion of active sentences in the training set). However, since the task includes passive structures, the network cannot decisively make any role assignments at this point. The indecision is manifested as oscillating activation values. The patient unit for the second noun block (row 2, column 2) also oscillates and competes with the recipient unit for the second noun block (row 3, column 2). When the verb "ate" is presented, the agent unit corresponding to noun 1 fires strongly since it is now clear that the sentence is not a passive construction. Similarly, the patient unit for noun 2 becomes more active since "ate" is transitive, and the recipient unit for noun 2 loses activation since "ate" does not take a recipient. The last part of the sentence further verifies the correct representation. This complex dynamic behavior emerged spontaneously. The units of the Structure level are predictive, but they postpone final decisions until sufficient information has been processed to make correct ones. The network's performance on a passive example sentence is shown in a later section.

3 Network Formalism

The task described above requires some special considerations in constructing a network formalism:

- Notions of time and sequentiality should be inherent.
- Units should be able to distinguish static and dynamic behavior.
- Atomic symbols should be represented and manipulated in a natural way.

The following defines a network formalism which has these properties. First, the runtime behavior of the network formalism is defined. Then, the derivation of the learning rule is outlined. Lastly, there is a short general discussion of symbol manipulation using this formalism.

3.1 Definition of Unit Behavior

A network is a collection of units which are interconnected arbitrarily. Time (denoted by t) is discrete, and units are updated synchronously. For each unit i, the following are defined:

$$\begin{aligned} o_i^t &\equiv \text{output} \\ v_i^t &\equiv \text{velocity} \\ a_i^t &\equiv \text{activity} \\ b_i &\equiv \text{bias} \\ d_i &\equiv \text{decay} \\ \Delta_i^t &\equiv \text{damping factor} \\ s_i^t &\equiv \text{stimulation} \end{aligned}$$

For each connection from unit j to unit i, there are two weights:

$$\begin{aligned} w_{ij}^o &\equiv \text{output weight to i from j} \\ w_{ij}^v &\equiv \text{velocity weight to i from j} \end{aligned}$$

At time $t = 0$, the velocity and activity of each unit is zero. The bias and decay of a unit are not time varying. The remaining quantities are calculated and updated as follows:

$$o_i^t = \sigma(a_i^t + b_i) \tag{1}$$

$$v_i^t = o_i^t - o_i^{t-1} \tag{2}$$

$$a_i^t = a_i^{t-1} d_i + \Delta_i^t s_i^t \tag{3}$$

$$\Delta_i^t = 1 - \left(\frac{a_i^{t-1} d_i}{M} \right)^2 \tag{4}$$

$$s_i^t = \sum_j \left(o_j^{t-1} w_{ij}^o + v_j^{t-1} w_{ij}^v \right) \tag{5}$$

$$\sigma(x) = 1 + e^{-\alpha x} \tag{6}$$

Equations 1 and 2 define the values of a unit i at time t which are externally available via output connections. Equation 3 shows how a unit behaves as it is updated. The activity is the sum of two terms: the residual activation and the damped stimulation. The residual activation is the activity remaining at time t from the activity at time $t - 1$. If the decay value is near one, most of the activity is retained; if the decay is near zero, little of the activity is retained. The stimulation comes from a unit's input connections. The damping factor prevents activation values from getting very large. Equation 1 adds in the bias term for the unit—essentially the resting activation value. The constant M is set large enough to allow the activity to be pushed into the flat region of the sigmoid, but not too far. This type of unit degenerates into a standard back-propagation unit if M is infinite, all w^v are zero, and the decay is zero.

When no stimulation is flowing into a unit, it decays towards a resting output value dependent on its bias. In response to positive stimulation over a sequence of time steps, a unit's activity gradually increases to a level dependent on the stimulation and the decay. The response to negative stimulation is analogous. Stimulation comes from two sources (see Equation 5): absolute output values/weights and velocity values/weights. Units are able to respond to *changes* in the activity of other units in addition to

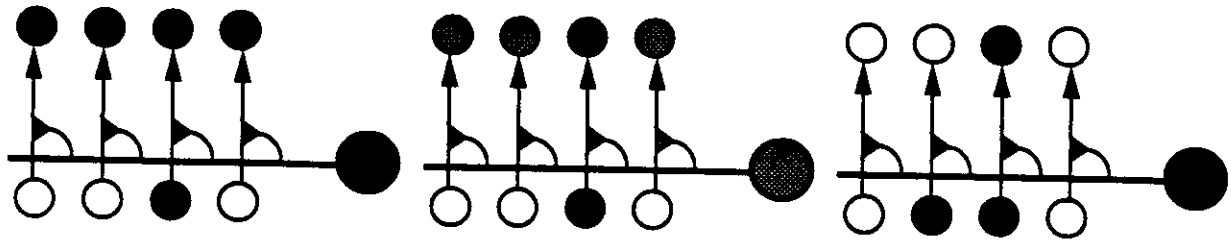

Figure 3: Slots of units with a gate.

the static output values. For example, a unit can learn to respond differently to another unit whose output value has been a steady extreme value and a unit whose output value just became extreme. It is not the case that in a trained network, wherever one finds a positive output weight, one finds a positive velocity weight. The signs can be opposing when a network learns complex dynamic behavior.

For the purpose of constructing symbol buffers from sets of units, units are allowed to *gate* connections between other units. A gate affects two things: the amount of stimulation flowing along the connections it gates and the decay values of the "to" units along those connections. When the gate is zero, no stimulation flows. When the gate is one, full stimulation flows. Partial stimulation flows with intermediate values. If a unit has input connections which are gated (possibly by different gates), its decay is modified by the gate whose value is maximal. When this value is one, the unit behaves normally. When zero, the unit's activity does not decay. Intermediate values scale the decay between the two extremes.

In Figure 3, imagine that the pattern of activation across the lower set of units represents a word. Assume that the connection weights are set such that the top slot takes on the activity pattern of the bottom slot when no gating unit is present. The gate functions as a control on assignment of the word in the bottom slot to the top slot. When the gate is on (has high output), the word gets assigned. When the gate is off, the word is "locked" into the top slot since no stimulation can flow nor can the units of the top slot decay. This type of assignment behavior would be difficult for a gradient descent network to learn to do robustly. By architecturally constraining a network using this type of device, one can manipulate symbol structures efficiently.

3.2 Learning

Learning is done using error back-propagation. In order to modify the behavior of a particular unit, one can modify two things: weights (including a bias term for each unit), and decay. Of these, the learning rule has been derived and applied only for the former. The decay values are set by the network designer according to criteria which are described later.[2] Details of the derivation have been omitted for the sake of brevity. The derivation essentially follows that given in [16].

An error measure for a network at time t is defined by:

$$E^t = \frac{1}{2} \sum_k (p_k^t - o_k^t)^2$$

where the sum is over all of the output units (those units which have explicitly defined target values p^t). The total error E for a particular input pattern is the sum over time of the individual E^t terms accumulated during presentation of the pattern.

It is necessary to minimize the error at each time step during a particular presentation. Thus, the

[2]There is no reason why decay values cannot be modified using gradient descent. This simply has not been explored yet.

central equation is given by:

$$\frac{\partial E^t}{\partial w_{ij}} = \sum_{s=1}^{t} \frac{\partial E^t}{\partial o_i^s} \left(\frac{\partial o_i^s}{\partial w_{ij}} - \frac{\partial o_i^s}{\partial o_i^{s-1}} \frac{\partial o_i^{s-1}}{\partial w_{ij}} \right) \tag{7}$$

The change in error at time t with respect to a particular weight (either output weight or velocity weight) is given by the sum over time of the change in error with respect to the output of unit i times the change in output at time s with respect to the weight less the change in output due to the change in output at time $s-1$. For an output unit, the error derivative can be calculated directly from the error measure. For a hidden unit, it can be calculated via back-propagation of error derivatives. The term inside the parentheses is easily calculated for any unit.

The equations defining the error derivatives with respect to the units' output values are most easily represented recursively. For an output unit i:

$$\frac{\partial E^t}{\partial o_i^t} = -(p_i^t - o_i^t) \tag{8}$$

$$\frac{\partial E^t}{\partial o_i^s} = \frac{\partial E^t}{\partial o_i^{s+1}} \frac{\partial o_i^{s+1}}{\partial o_i^s} \tag{9}$$

Equation 9 shows how error derivatives are propagated back through time for an output unit. The first term is known. The second term is easily derived from Equations 1 through 6 for any unit. For hidden units, the situation is more complex. The change in the error due to an infinitesimal change in o_j^s has two components:

$$\frac{\partial E^t}{\partial o_j^s} = \frac{\partial E^t}{\partial o_j^{s+1}} \frac{\partial o_j^{s+1}}{\partial o_j^s} + \sum_i \frac{\partial E^t}{\partial o_i^{s+1}} \frac{\partial o_i^{s+1}}{\partial o_j^s} \tag{10}$$

where the sum is over those units i which have input connections from j. The first term is the change in error arising from a change in the next output value of j. The second term is the change in error arising from the response of the units i to the change in their common input unit. The error derivatives for the units i are calculated recursively starting from the output units (as in standard back-propagation). The other terms in Equation 10 are easily calculated. Having calculated the change in error at time t with respect to the output of a unit at times $s = 1$ to $s = t$, it is an easy matter to calculate the remaining terms of Equation 7. This gives the change in error at time t with respect to a particular weight.

The process by which weight changes are calculated appears to be rather complex just looking at the equations. A walk through one time step will be instructive. At each time step, the values of the various terms in the equations which are not dependent on the values of the targets are calculated and stored. After all activities, output values, and velocity values have been calculated for one time step, the target values for the output units are made known to the network. From these, the error derivatives with respect to the output values are back-propagated through the connections of the network and through time. From here, it is just a matter of summing up the various terms to calculate the error derivative for each weight in the network. These are accumulated for each time step until presentation of the input is terminated. Then, the weights are modified so as to reduce the error terms at each time step. The implementation allows for full recurrence in the connectivity of networks.

A word about "output" units is in order. Those units in a network which have modifiable input connections and for which the teacher of the network has particular values in mind are output units. This includes gating units which control the behavior of units which have target values. Units whose connections are gated do not have modifiable connections and are not considered to be output units. The behavior of such units is defined by the gates along their input connections; the units themselves serve

as storage buffers. Appropriate targets for the gates are calculated at each time step in accordance with the external targets provided for the units which are affected. The weights are modified so as to closely approximate the ideal dynamical behavior of the gates.

Earlier, it was mentioned that decay values for units are chosen by the network designer and are not modified. Modification of these values by gradient descent is something which should be explored, but it has not been necessary in the types of experiments which have been attempted using this formalism thus far. When one constructs a symbol buffer using a set of units with a gate (see Figure 3), a few factors influence the choice of decay values for the units. In order for a symbol buffer to behave reasonably, the gating unit responsible for modulating the input to the buffer must do three things:

- Become active when an appropriate symbol is present on the units which feed the buffer.
- Remain active at least until the symbol is stored in the buffer.
- Become inactive and remain so (unless a new symbol is to be stored in the buffer).

If the decay for the symbol buffer units is set such that activation falls slowly and the weights on the input connections to the buffer are quite small, then the gating unit for the buffer can be relaxed about becoming active and inactive as long as the symbols remain available for a number of time steps. Conversely, if the weights are high and the decay is fast, the gating units must respond very quickly to changing stimuli to avoid missing a symbol or assigning it and allowing it to either decay or become corrupted by another incoming symbol. The symbol buffer must be constructed with the expectations of its dynamical environment in mind.

The decay values for other units are set so as to allow smooth integration of new information but reasonably quick response to changing information. An implementation issue also is involved in the selection of acceptable decay values. Recall that in Equation 7, the sum of terms for calculating weight changes is supposed to begin at the start of the current input pattern presentation. This would present formidable time and space complexity if implemented exactly. To partially alleviate this, the sum is only calculated for a few time steps back from the current one. This is a very good approximation if all of the units with modifiable input connections decay quickly enough so as to make the contribution of any changes from early in the time sequence negligible.

3.3 Manipulating Sequential Symbols

Systems in domains such as natural language processing must manipulate symbolic objects. If the input to a network is symbolic, and the desired output of the network consists of some transformation of the input which preserves the integrity of the symbols, then it makes sense to constrain a network to treat symbols atomically. Further, if it is required that the input be presented sequentially, it is desirable to have stable storage of symbols and to have a network which is able to smoothly adapt to new information as well as react to its own evolving output representation. It is possible to construct such networks using the formalism described in the foregoing sections.

One can legitimately ask where the win is if all that is happening is symbol manipulation. The answer lies in the way the "rules" are encoded. Instead of being highly structured symbolic rules such as those found in grammars, networks learn their behavior by learning *soft* rules which may not have any absolute conditions. Another aspect is that a network makes no fundamental distinction in the way symbolic and non-symbolic information is represented—everything is patterns of activation. It becomes possible to combine multiple modalities of information in the input of a network and train the network to respond appropriately to the portions of the input which have meaningful content. The hope is that this type of learning will generalize well and be robust in the face of noise.

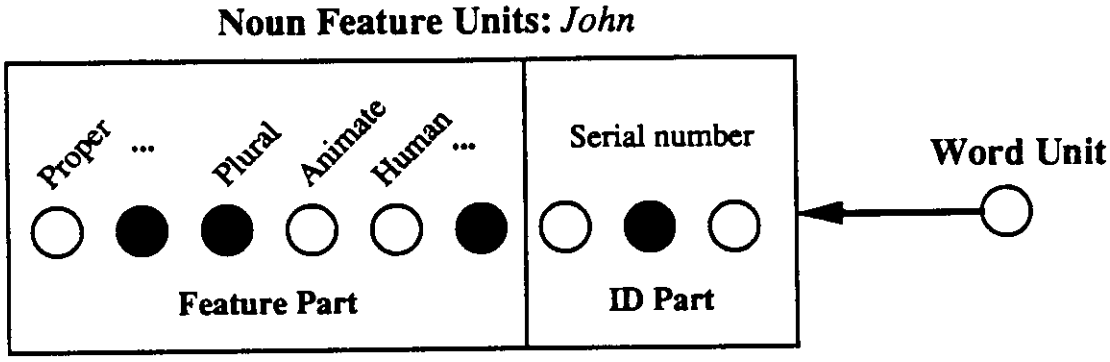

Figure 4: Word Representation.

4 Parsing Network Architecture

The parsing architecture is modular, recurrent, and hierarchical. An overview of the parsing architecture was given in Section 2.2. This section will proceed bottom up from the Word level to the Structure level and give the details of the architecture which were omitted earlier. Then the procedure used to construct and train the parsing network will be explained.

4.1 Word Level

The meanings of words are represented as microfeature patterns much the same way as in [13]. The representations are not learned. In [13], the microfeature patterns formed the input to a network. In this architecture, each word in the lexicon is assigned to a single unit in the network, and the microfeature representation of a word is encoded in fixed output connections from the unit corresponding to the word. Words are divided into seven classes: nouns, verbs, adjectives, adverbs, auxiliaries, prepositions, and determiners. Each class has a set of feature units with fixed connections from word units. There are two parts to a word's representation: the feature part and the identification part, as shown in Figure 4. The feature part contains both syntactic and semantic information. The identification part serves to distinguish between words which have identical feature parts (e.g. Peter and John). The units in the network which learn do not have any input connections from the identification parts of words. This has the very convenient effect of allowing new individuals to be added to a network without additional training as long as they are not syntactically/semantically distinct from old ones.

A word is presented to the network by artificially stimulating its associated word unit. This causes the representation for a word to appear gradually during the course of a few time steps across one (possibly more) set of feature units. The pattern of activation across a single set of feature units constitutes a symbol which should not be modified during further processing; however it may be moved around from slot to slot.

It should be noted that it is possible for connectionist networks to learn internal representations of words from their usage in sentences [14]. However, the focus of this task is on dynamic parsing behavior, not on acquisition of lexical information. Also, in more realistic domains with larger vocabularies, some degree of prewiring lexical information is a pragmatic design choice for the construction of large-scale practical systems.

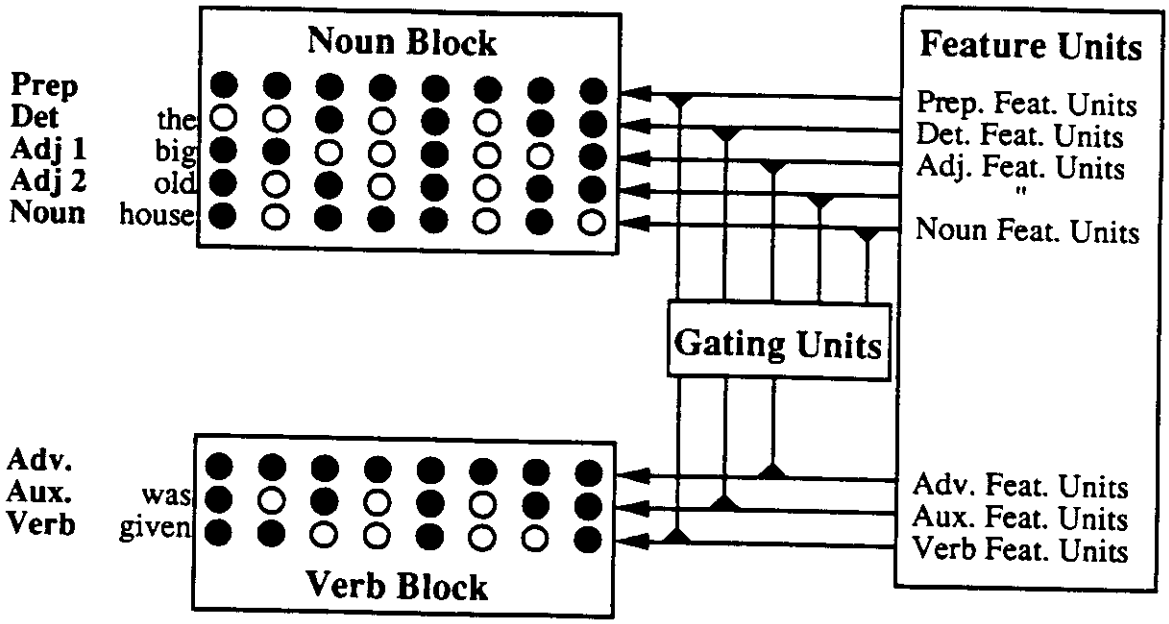

Figure 5: Phrase Representation.

4.2 Phrase Level

Phrases are represented as head words with modifiers (called blocks). For example, "the big old house" would be represented as the noun "house" modified by the determiner "the" and the adjectives "big" and "old". Figure 5 shows the structure of noun and verb blocks. A noun block has slots for a noun, two adjectives, a preposition, and a determiner. A verb block has slots for a verb, an auxiliary, and an adverb. Each word slot of each block has fixed connections from one set of feature units. The connections for each slot share a single gating unit (see Figure 3). These gating units completely control the behavior of the slots in the blocks. When the gating unit for a particular slot is active, the pattern of activation on the feature units becomes active on the units of the slot.

Figure 6 depicts the network's detailed structure. In the diagram, the units shown in thick lined boxes have modifiable input connections—they learn their behavior. The gating units at the Phrase level share a group of hidden units. These hidden units have connections from the feature units, the noun and verb blocks, and the gating units themselves. The Phrase level forms a recurrent subnetwork. It is recurrent in two ways. One, there is direct recurrence arising from the connections among the gates and the hidden layer. Each has modifiable weights to the other. Two, the hidden layer has input connections from the noun and verb blocks which are completely controlled by the gating units. Both types of recurrence are important in the behavior of the gating units.

4.3 Structure Level

The units in the Structure level describe the relationships between the phrases in the Phrase level the clauses they make up. There are six relationships possible (see Figure 6). If unit at position X,Y of a (rectangularly organized) group is active, it means the following:

- Agent Group: Noun block (NB) X is the agent of verb block (VB) Y. Group of 3 by 2 units.
- Patient Group: NB X is patient of VB Y. Group of 3x2.

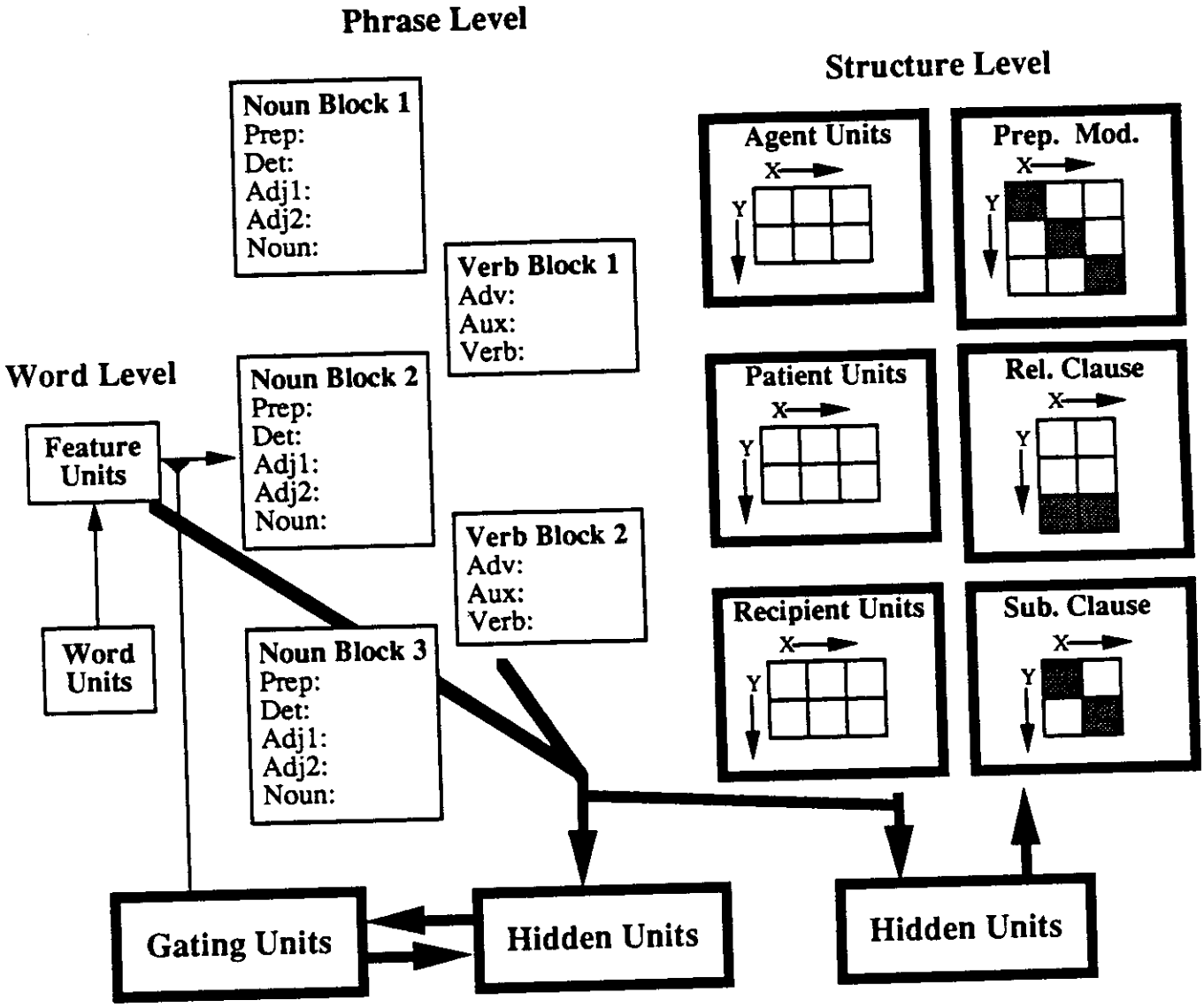

Figure 6: Detailed Network Architecture.

- Recipient Group: NB X is recipient of VB Y. Group of 3x2.
- Prepositional Modification Group: NB X modifies other NB Y. Group of 3x3.
- Relative Clause Group: VB X modifies NB Y. Group of 2x3.
- Subordinate Clause Group: VB X subordinate to VB Y. Group of 2x2.

The units of the Structure level also share a set of hidden units. These hidden units "see" all that the other set of hidden units see but have connections from the structure representation units instead of connections from the gating units. The Structure level forms a recurrent subnetwork as did the Phrase level.

4.4 Training

A network with this architecture is trained in two phases. First, the gating units in the Phrase level which are responsible for the behavior of the slots of the noun and verb blocks are trained. Their behavior is quite complex. They must learn to turn on when a word appears across the feature units for their slot (and their slot is supposed to be filled), stay on until the word disappears (even after the word has been assigned to the slot), turn off sharply, and stay off even when another word appears across their feature units. They must also learn to overwrite or empty out incorrectly assigned slots. Words get assigned incorrectly when they have representations in more than one class and there is insufficient information to disambiguate the usage. The word "was" has representations both as a verb and as an auxiliary verb. The network must assign it to both the auxiliary and the verb slots of the current verb block, and disambiguate the assignment when the next word comes in by either overwriting the verb slot with the real verb or emptying out the auxiliary slot.

The next phase involves adding the Structure level and training the structure representation units. The targets for the structure units are set at the beginning of a sentence and remain the same for the whole sentence. This forces the units to try to make decisions about sentence structure as early as possible; otherwise, they accumulate error signals during the initial part of the presentation of the input pattern. An interpretation is considered successful if the structure units are in the appropriate states at the *end* of the input. On the surface, it may seem that these units should have more or less monotonic behavior. However, the sentences in this domain do not necessarily contain sufficient information at word presentation time to make accurate decisions about the word's function. This coupled with the network's attempt to make decisions early causes the structure units to have surprisingly complicated activation patterns over time.

A set of nine sentences was used to train the gating units of the Phrase level of a network. They were selected to be the smallest set of sentences which would cover a reasonably rich set of sentences for training the Structure units. The network generalized very well to include "compositions" of sentence types from the initial set of nine. From this network, the Structure units were added. Eighteen sentences which were correctly processed at the Phrase level were chosen to train the Structure level. A variety of sentences was included. There were more active constructions than passive, more single clause than two clause sentences. Many different role structures were present in the training set. The network learned the set successfully. In what follows, references to "the network" or to substructures of a network are referring to the network described here.

5 Parsing Network Performance

It should be noted here that the goals of this task are to see if:

- The network architecture "works".
- It can learn intricate dynamic behavior.
- It can learn to incorporate previously unseen right context into an evolving representation.
- It generalizes.
- It is robust with respect to noisy ill-formed input.

The goal is not to posit a theory of human language processing. There are three aspects of performance which will be characterized. First, the dynamic behavior of the network will be discussed. The ability of the network to generalize will follow. Lastly, network performance on noisy input will be explored.

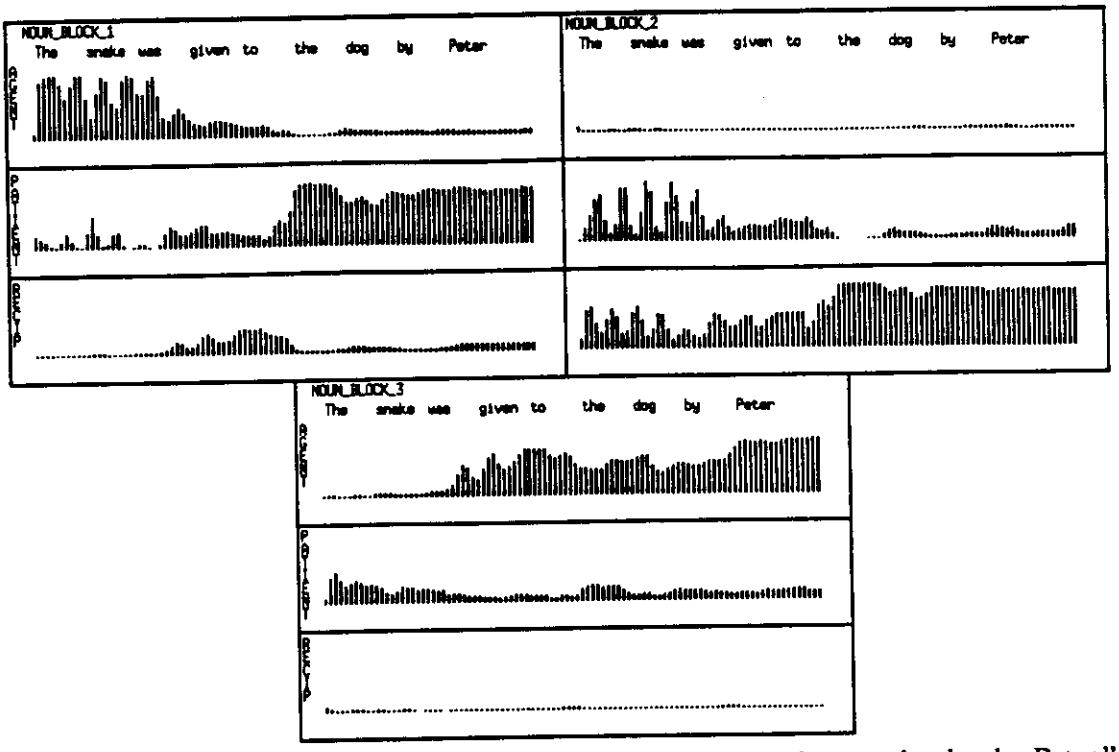

Figure 7: Performance of the network on, "The snake was given to the dog by Peter."

5.1 Dynamic Behavior

The units of the Structure level in the trained network display several interesting properties during the parsing process:

- They attempt to be predictive.
- They quickly respond to violations of predicted behavior when incorporating new right context information.
- They oscillate during periods of uncertainty.
- They show strong competitive effects which are learned via the recurrent connections.

These aspects of the network's behavior will be illustrated with some example sentences.

The performance of the key Structure level units on, "A snake ate the girl," was shown in Figure 2. On this simple sentence, the network's dynamic behavior was still somewhat complex. The role representation units were predictive but postponed final decisions until they could be made with confidence. Periods of indecision were marked by oscillations among competing units. Ultimately, the expectations of the network were confirmed.

The behavior of the network on passive sentences is more involved. A passive sentence with Patient/Recipient/Agent role structure is shown in Figure 7. The behavior of the units is similar to the previous sentence in the beginning. When the passive construction is detected, several things happen. The agent unit for noun 1 loses activity. The recipient and patient units for nouns 1 and 2 show modest activity—anticipating the likely possible role structures. Also, the agent unit for noun 3 anticipates the distant terminal phrase. When the preposition "to" is presented, the units quickly respond and indicate

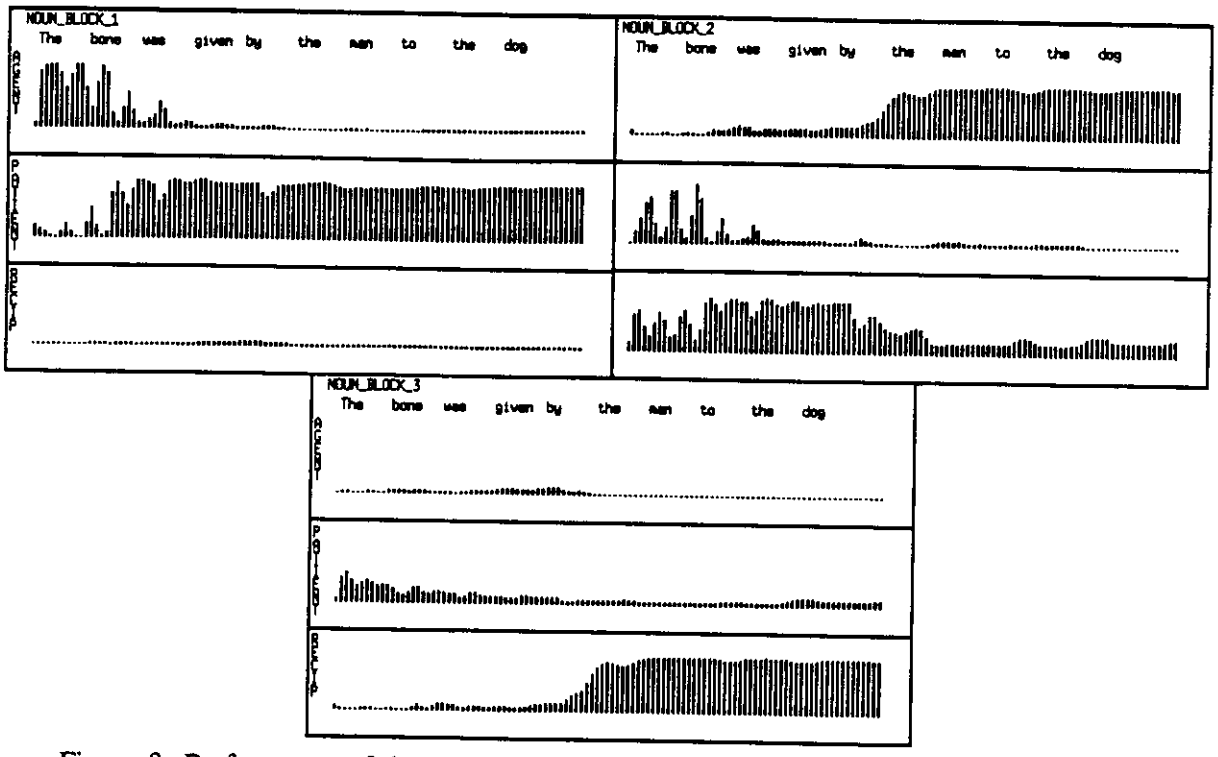

Figure 8: Performance of the network on, "The bone was given by the man to the dog."

that noun 1 is the patient and noun 2 is the recipient (see points A and B in Figure 7). The lingering activation of the agent unit for noun 3 finally becomes most active when the phrase "by Peter" confirms the unit's expectation. The role structure is assigned correctly.

This strongly predictive behavior would be undesirable if it were not possible to train the network to respond correctly to exceptions in generally prevalent structures. One sentence with a Patient/Agent/Recipient role structure was present in the training set. Figure 8 shows the network's behavior on that type of sentence. During the early part of the sentence, the network detects the passive construction, and the agent unit for noun 1 loses activation. Since, "bone" is inanimate, the patient unit for noun 1 is able to quickly assign "bone" the role of patient. It quickly achieves a steady high activation, and the competing patient unit settles into a steady low activation. However, the network expects noun 2 to be the recipient since that is the most common construction in the training set. The preposition "by" violates the default expectation of the network, and the units adjust accordingly (see A in Figure 8). The recipient units for nouns 2 and 3 invert their respective activities (points B and C). The agent unit for noun 2 quickly responds. The network was able to learn to react quickly to this sharp exception to the network's default expectations. The competing units behave sensibly, showing partial or oscillating activation for expectations, and responding quickly when right context information disambiguates the role assignment.

5.2 Generalization

The previous section discussed dynamic behavior of the network on sentences which it "expected" to see, but a very important aspect of network behavior is its ability to generalize to include well formed sentences which are *not* explicitly learned among those sentences which can be successfully processed. Two types of generalization will be discussed here: automatic generalization, and generalization to truly novel well formed sentences.

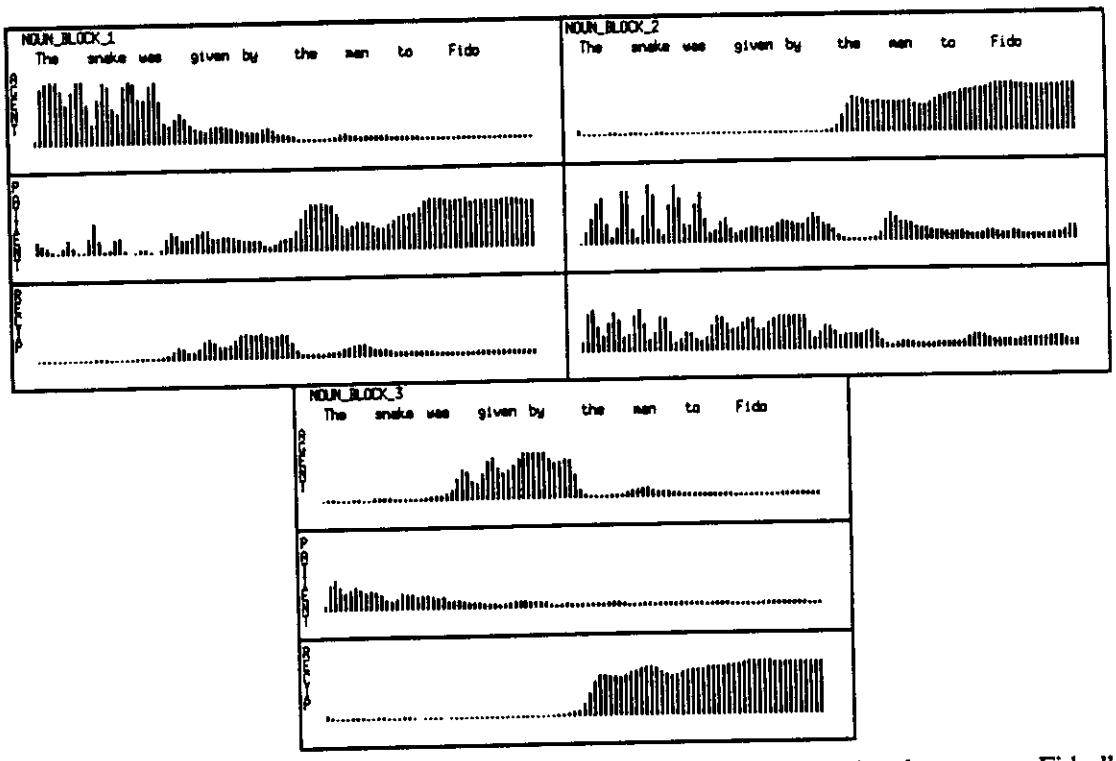

Figure 9: Performance of the network on, "The snake was given by the man to Fido."

Automatic generalization is built into the network through the representation of words. For example, if the network successfully learns to parse "Peter gave the bone to the dog," it will know how to parse "Jim promised the mitt to the boy." This may appear to be a mere trick, but if one really wants to build connectionist networks to operate over domains with large vocabularies, techniques such as this which minimize training costs while producing better generalization will be useful. This type of generalization is important, but isn't particularly surprising—it is built into the network.

The other type of generalization is where a novel sentence is processed correctly by the network. A novel sentence is any sentence which is not isomorphic to a sentence in the training set modulo the identification bits of the words in the sentence. Generalization to novel sentences is clearly important if one hopes to have manageable training sets. One cannot expect the network to generalize to the point where it would correctly process a sentence with a role structure not present in the training set. However, if the network is able to tolerate variations in phrase structure and variations in the features of words, the necessary training set can be greatly reduced. Since the task is very restricted, it is difficult to measure this type of generalization meaningfully; however some examples will be instructive.

Figure 9 shows a variation of the sentence shown in Figure 8 which is similar to the sentence in Figure 7. Recall that only one sentence with a Patient/Agent/Recipient role structure was in the training set. The sentence, "The snake was given by the man to Fido," differs from the training sentence in two respects: "snake" is animate and the terminal noun phrase has a proper noun. The fact that "snake" is animate is quite significant since it is possible for animate nouns to be recipients. If the network simply memorized the single exceptional exemplar from the training set, this new sentence would not be processed properly. Figure 9 shows that the network behaves correctly. Initially the structure units behave as they did in Figure 7. The expectation is for noun 3 to be the agent. There is considerable uncertainty about the role of noun 1 since it can be either a patient or a recipient (unlike the sentence in Figure 8). When the phrase "by the man" is presented, the network reacts quickly (see the indicated points in the

diagram) and ultimately assigns the correct role structure although the path to the assignment is different from the path taken on the training sentence. In this example, the network treats the novel input as it would its closest exemplar from the training set.

A more difficult situation occurs when the most similar sentence from the training set does *not* have the correct role structure. Rather, the detailed features of the words must dominate the role assignment. The sentence, "A snake was given an apple by John," was in the training set. It has a Recipient/Patient/Agent role structure. A similar sentence given by, "A bone was given the dog by John," receives a Patient/Recipient/Agent role structure by the network, as is desired. Here, the network must combine several different types of evidence to come up with the correct answer. It appears to be expressing a heuristic such as "Inanimate objects are preferred as patients over animate ones." Soft rules like this are combined with other knowledge about word order and sentence structure to parse novel sentences.

5.3 Noise

In spoken language systems, there are many sources of noise. Two prominent ones are potential errors in the incoming word hypotheses and the user. Since it is not possible to achieve perfect performance from either the speech recognition system or the speaker, the higher level systems must be able to tolerate such problems.

In a tightly coupled system where the parsing level feeds back to the speech level, variations in word duration must not degrade performance. Despite being trained using a constant word duration, the network handled a thirty percent variation in duration with no performance loss. Inter-word silences also had no effect.

Determiners and other short function words often are poorly articulated. This is a persistent problem for speech recognition systems as it leads to word deletions. Despite such deletions, the network makes appropriate role assignments with such sentences as "Snake ate girl." The role assignment is Agent/Patient as in the uncorrupted sentence. Random deletion of determiners from sentences does not lead to serious misinterpretation (an incorrect role assignment) ninety percent of the time.

Non-speech interjections are also possible as in, "A snake (ahh) ate the girl." A speech recognition system can easily interpret the non-speech "ahh" as "a". In this case, the network puts the non-speech "a" in the determiner slot of the second noun block, and then overwrites it with "the". The result is a good parse of the ill-formed sentence. Interjections of "ahh" before verb phrases do not interfere with proper role assignments. Some errors in the noun phrases do occur, but all content words are stored in the correct blocks.

Multi-word repetition in noun phrases such as, "John gave a bone to the to the dog," occur in spontaneous speech. The network responds most often by re-assigning the repeated portion of the phrase to the appropriate block. This has the nice effect that sentences like, "John gave a bone to the ... to a dog," are usually handled by overwriting the old phrase with the modified one—a desirable effect. Also, ungrammatical sentences such as, "We was happy," are generally treated by the network in the same manner as their closest grammatical counterpart.

5.4 Summary

It is important to note that this domain is so small that these results can only be suggestive of potential performance in larger, more realistic domains where performance is easier to quantify. However, the goals of the experiment have been met. The network was able to successfully learn to combine syntactic, semantic, and word order information effectively. The network displays complex dynamic behavior and is able to confirm or revise hypotheses based on right context information. It generalizes well and is robust with respect to noise.

6 Extensions

With the encouraging results obtained from this experiment, an experiment much larger in scope is being attempted. The domain is sentences with up to three clauses. The training corpus contains over 200 sentences including sentences with passive constructions and complex center embedded clauses. A network is being trained to do phrase parsing, role assignment, relative and subordinate clause identification, and prepositional attachment. Intermediate results on such a network have been promising.

Future work will focus on building real spoken language systems using networks of this type. Networks which provide feedback to word recognition systems for perplexity reduction will be explored. Networks which incorporate prosodic information at the parsing level are also of interest.

7 Conclusion

Connectionist formalisms which have the ability to manipulate symbols through a time sequence show promise in helping to solve some difficult problems in spoken language processing and other domains. There are three central strengths of the connectionist approach. Networks learn their behavior. The rules which are learned are not rigid or formal and hence tend to generalize and be resistant to noise. Networks can learn to incorporate information from multiple knowledge sources and input modalities.

Acknowledgments

I thank Alex Waibel and Dave Touretzky for useful comments during the evolution of this research and during the preparation of this report.

References

[1] E. Charniak and E. Santos. A connectionist context-free parser which is not context-free but then it is not really connectionist either. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pages 70–77, 1987.

[2] G. W. Cottrell. *A Connectionist Approach to Word Sense Disambiguation.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[3] G. W. Cottrell. Connectionist parsing. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pages 201–211, 1985.

[4] G. W. Cottrell, P. W. Munro, and D. Zipser. Image compression by back propagation: A demonstration of extensional programming. In N. E. Sharkey, editor, *Advances in Cognitive Science*, Ellis Horwood, Chichester, England, 1987.

[5] J. L. Elman. *Finding Structure in Time.* Technical Report 8801, Center for Research in Language, University of California, San Diego, 1988.

[6] J. L. Elman and D. Zipser. Discovering the hidden structure of speech. *Journal of the Acoustical Society of America*, 9, 1988.

[7] M. Fanty. Context-free parsing with connectionist networks. In J. S. Denker, editor, *AIP Conference Proceedings number 151*, American Institute of Physics, New York, 1986.

[8] S. J. Hanson and J. Kegl. PARSNIP: A connectionist network that learns natural language grammar from exposure to natual language sentences. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pages 106–119, 1987.

[9] T. Howells. VITAL: A connectionist parser. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, pages 18–25, 1988.

[10] A. N. Jain and A. H. Waibel. A connectionist parser aimed at spoken language. In *Proceedings of the International Workshop on Parsing Technologies*, pages 221–229, Carnegie Mellon University, Pittsburgh, PA, August 1989. Available through the School of Computer Science, Carnegie Mellon University.

[11] F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, April 1976.

[12] M. I. Jordan. *Serial Order: A Parallel Distributed Processing Approach.* Technical Report 8604, Institute for Cognitive Science, University of California, San Diego, 1986.

[13] J. L. McClelland and A. H. Kawamoto. Mechanisms of sentence processing: Assigning roles to constituents. In J. L. McClelland and D. E. Rumelhart, editors, *Parallel Distributed Processing*, The MIT Press, 1986.

[14] R. Miikkulainen and M. G. Dyer. Encoding input/output representations in connectionist cognitive systems. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 347–356, Morgan Kaufmann Publishers, 1989.

[15] M. Nakamura and K. Shikano. A study of English word category prediction based on neural networks. In *Proceedings of the 1989 IEEE International Conference on Acoustic, Speech, and Signal Processing*, pages 731–734, May 1989.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, The MIT Press, 1986.

[17] B. Selman and G. Hirst. A rule-based connectionist parsing system. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pages 212–221, 1985.

[18] D. Waltz and J. Pollack. Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science*, 9:51–74, 1985.