Parallel Distributed Processing: Bridging the Gap Between Human and Machine Intelligence

James L. McClelland, Axel Cleeremans, and David Servan-Schreiber
Carnegie Mellon University

Abstract

The Parallel Distributed Processing approach to modeling intelligent information processing grew out of the feeling that conventional computational approaches to modeling intelligent information processing were not well suited to capturing several key aspects of human information processing abilities. In this paper, we describe these key aspects of human processing abilities, and we note how they can be captured in parallel-distributed processing models (otherwise known as connectionist models or neural network models). A key feature of these models is their ability to make effective use of *graded* or *continuous-valued* information. We then note how the ability to make use of graded information is exploited in several current connectionist AI research projects. The final section of the paper illustrates how machines that make use of graded infromation may increase our taxonomy of basic computational machine types. We introduce the notion of the *graded state machine* and argue that it has characteristics which place its capabilities between finite-state and recursive computational devices.

Parallel Distributed Processing: Bridging the Gap Between Human and Machine Intelligence

James L. McClelland, Axel Cleeremans, and David Servan-Schreiber
Carnegie Mellon University

Abstract

The Parallel Distributed Processing approach to modeling intelligent information processing grew out of the feeling that conventional computational approaches to modeling intelligent information processing were not well suited to capturing several key aspects of human information processing abilities. In this paper, we describe these key aspects of human processing abilities, and we note how they can be captured in parallel-distributed processing models (otherwise known as connectionist models or neural network models). A key feature of these models is their ability to make effective use of *graded* or *continuous-valued* information. We then note how the ability to make use of graded information is exploited in several current connectionist AI research projects. The final section of the paper illustrates how machines that make use of graded infromation may increase our taxonomy of basic computational machine types. We introduce the notion of the *graded state machine* and argue that it has characteristics which place its capabilities between finite-state and recursive computational devices.

Parallel Distributed Processing: Bridging the Gap Between Human and Machine Intelligence

James L. McClelland, Axel Cleeremans, and David Servan-Schreiber
Carnegie Mellon University

Every year computers get faster; yet human beings remain so much better than computers at performing a wide range of basic, natural, cognitive tasks, such as speech recognition, language understanding, and retrieval of contextually appropriate information from memory. The reason for this is that conventional computer architectures are not well suited to these natural cognitive tasks.

In this article we will point out several characteristics of human cognitive processes that conventional computer architectures do not capture well. Then we will note that connectionist models are much better able to capture these aspects of human processing. After that we will mention three recent applications in connectionist artificial intelligence which exploit these characteristics. Thus, we shall see that connectionist models offer hope of overcoming the limitations of conventional AI. The paper ends with an example illustrating how connectionist models can change our basic conceptions of the nature of intelligent processing.

Characteristics of Human Information Processing

There are three essential characteristics of human intelligence that are not well captured by conventional computer architectures:

- Use of multiple, graded constraints.
- Use of conspiracies of multiple memory traces.
- Sensitivity to soft as well as hard regularities.

There is a lot of psychological evidence for each of these points. Some of this evidence is described in the following paragraphs.

--- INSERT FIG 1 ABOUT HERE ---

Use of multiple, graded constraints. In perception, we use context as well as stimulus information. Thus to identify a letter in a word we use the features of the letter, together with information from context. Similarly, to identify a drawing of an object, we use the features from the drawing and contextual information. For example, Labov (1973) showed human subjects drawings of objects that vary in several properties. A set of drawings he used is shown in Figure 1. He found that most people called the one in the middle of the top row a cup. The probability of calling the object a cup drops as the drawing becomes tall or wide, and also drops off when the handle is removed, but *both kinds of changes have a graded effect.* Furthermore, the probability of calling an object a cup can be affected by context; if you put a flower in one of the pictures people are less likely to call it a cup; if you put it next to a teapot the probability of calling it a cup goes up. This example illustrates that each source of information exerts a graded influence and many sources operate together.

Use of conspiracies of mental agents. This point is similar to the last one, but not exactly the same. When people must categorize stimuli, it appears that they generally use memory traces of many similar stimuli to do the categorization. The evidence does not fit theories that say that they extract a single prototype or general rule; rather, the evidence fits much better to the theory that they form memory traces of each example seen, and when they are asked to categorize a new example, *all previous examples exert an influence,* proportional to the similarity to the new example. Sometimes, the result of this is that the prototype, or central tendency of the set of examples, produces the strongest response, even though the prototype has not actually been seen. This occurs when many examples similar to the prototype have been seen. In any case, though, it can be shown that there is also sensitivity to particular examples that have been seen as well (See McClelland and Rumelhart, 1985,

for a discussion).

Exploitation of soft vs. hard regularities. Conventional computer systems are good at operating by rules, but have difficulty with situations in which there are "soft" regularities. For example, in English there is a "regular" past tense, formed by adding "ed" to the base form of a word (e.g., look -> looked). Most conventional systems treat all other past tenses as exceptions. However, there are many groups of words that have a similar way of forming the past tense. There are many no-change verbs (hit, cut, put ...), for example. The past tense of these words is the same as the present tense. These words are all one-syllable, they all end in d or t, and most have a short vowel. However, some words like this do not fit the rule (sit->sat, rid->ridded). We know from experiments that people can exploit these "partial" or soft regularities; they seem to treat them in a way that is intermediate between regular and unique exception words. In conventional systems these intermediate cases are difficult: we must choose between writing a rule or not. If we write a rule we have the danger of making mistakes for some words that do not fit the rule. If we do not, we are in danger of missing the generalization.

Connectionist Models

Connectionist models are well suited to capturing these characteristics of human processing capabilities. In connectionist models, processing takes place through the propagation of graded activation signals among a set of simple processing units through connections of graded strength. Considering the three points raised above:

- Multiple graded influences can be captured simply by the convergence of multiple inputs on a single processing unit or set of units. Since the activations and the weights are all real-valued, each input exerts a graded influence on the outcome of processing.
- Conspiracies of memory traces are similarly very easy to capture; each "memory trace" can be represented by a single unit that is activated by an input by an amount that increases with its

similarity to the input, and contributes to the output in proportion to its activation.

- The exploitation of soft as well as hard regularities occurs in several ways in connectionist models. It can occur through the conspiracy mechanism just described, or through the influence that each example of a regularity has on the values of the connection weights in a system that is trained on a set of examples.

The framework for building connectionist models is laid out in detail in Rumelhart, McClelland and the PDP Group (1986), and many examples of models constructed in that framework are described.

Two examples of connectionist models of human processing abilities that capture these characteristics are the interactive activation model of visual word recognition from McClelland and Rumelhart (1981), and the model of past tense learning from Rumelhart and McClelland (1986). These models were motivated by psychological experiments, and were constructed to capture the data found in these studies. We describe them here to illustrate some of the roots of the connectionist approach in an attempt to understand detailed aspects of human cognition.

--- INSERT FIG 2 ABOUT HERE ---

The Model of Word Recognition

The model of visual word recognition, shown in Figure 2, made use of units at visual feature, letter, and word levels to process strings of letters. Connections ran bottom-up, from the feature to the letter level, and then from the letter to the word level; and top-down, from the word to the letter level and from the letter to the feature level. The connections allow the word and feature levels to simultaneously constrain the activation of units at the letter level. There are also mutually inhibitory connections within levels, to allow the unit or units receiving the strongest activation to suppress others which might otherwise become activated. When an input is presented that contains an

ambiguous letter as in Figure 3a, the bottom-up information alone is not sufficient to determine the identity of the letter. In this case all the possibilities become gradually activated. But at the same time, the information about the other letters is propagating forward. If (as in the example illustrated in Figure 3a) the letters the the other positions go together with only one of the alternatives in the position containing the ambiguity to form a word, the unit for this word will become most active. The top-down connections will then cause the letter that fit together to make this active word to become more active that the other letters in the ensemble. The model reaches this result as a result of gradually adjusting activations to settle into a "good" state, in which the activations at all levels are maximally consistent. In other simulations, it is shown that this graded constraint satisfaction process can account for a wide range of experimental findings on the combination of graded sources of information such as those illustrated in Figure 1 above.

--- INSERT FIG 3 ABOUT HERE ---

The model also exhibits the "conspiracy" of multiple units at the same level. Thus, when a non-word such as Y?AT is shown (Figure 3b), the input in the second position is consistent with E, C, or F in the particular type font used in the simulation. In this case there is no single word that matches one letter from each position; but many words are partially activated. Some are illustrated in Figure 3b. Another word that is partially activated is the word YEAR, for example. As it happens, there are many words that have an E in the second position that are partially activated since they match two of the three unambiguous letters as well as the E in the second position; but there are none that have either a C or F in the second position that become partially activated. All of the partially activated words feed back activation to the letter level, thereby causing the unit for E to receive more activation and win out in the competition with other units.

--- INSERT FIG 4 ABOUT HERE ---

The Model of Past-tense Formation

The word perception model just reviewed suggested that a simple connectionist network could behave *as though* it knew the rules that govern what makes a sequence of letters count as a legal letter string in English. This kind of characteristic is also seen in connectionist models that learn from a set of training examples. One such network is the model of past-tense learning studied by Rumelhart and McClelland (1986), shown in Figure 4. The network was trained by showing it input patterns representing the present tenses of words. At first the connections in the network were all 0's, so that output was uninteresting. However, an error correction learning rule was used to adjust the connections from input units to output units so as to reduce the difference between the network's actual output and the correct output. Gradually, the network came to be able to produce the past tenses, not only of words that it actually was trained on, but also of novel words. for most novel words, the model produced the correct past tense, according to the standard past tense rules of English. However, for novel words that were similar to known exceptions, it tended to transform the input in a way that was similar to the handling of the exception. This tendency was particularly strong for words that belonged to clusters of exceptions. Thus, the model exhibited a sensitivity to the "soft" or partial regularities present in the training set as well as to the "hard" or general rule of English past tense formation.[1]

---

[1]The Rumelhart and McClelland paper has been criticized by Pinker and Prince (1988) for failing to capture the general rule strongly enough and for other shortcomings. However, most of these shortcomings have been addressed and overcome in a recent improvement on the original model by Plunkett and Marchman (1989).

Connectionist models in Artificial Intelligence

The models just reviewed capture important aspects of data from psychological experiments, and illustrate how the characteristics of human processing capabilities enumerated above can be captured in an explicit comptutational framework. Recently connectionist models that capture these same characteristics have begun to give rise to a new kind of Artificial Intelligence, which we will call connectionist AI. Connectionist AI is beginning to address several topics that have not been easily solved using other approaches. We will consider three cases of this. In each case we will describe recent progress that illustrates the ability of connectionist networks to capture the characteristics of human performance mentioned above.

All of the examples in connectionist AI rely on a procedure known as back-propagation to adjust the strengths of the connections among units so as to reduce the error between the output of a network and the correct or target output. Interestingly, back-propagation works only in networks consisting of units that take on graded activation values. This is because back propagation requires the existence of a smooth derivative of the activation of each unit with respect to its input from other units. Only in this way can one calculate connection changes that adjust weights in the network in the direction of reducing the error. A detailed description of the back-propagation learning procedure is described in Rumelhart, Hinton and Williams (1986).

Phoneme recognition. The problem of phoneme recognition has been a notoriously difficult problem since researchers first tried to identify features of speech that contributed to phoneme identification. Though various features may be present in the sound-spectrograph, tokens of phonemes are highly context sensitive; features may be distorted or absent, depending on context, rate of speech, etc. Many systems that use graded information have been developed in the last few years; some are fully connectionist and some are not. Right now it appears that one of the most successful systems is a connectionist system, the Time-Delay Neural Network (TDNN) (Lang, 1987; Waibel, 1989).

The TDNN is a conventional connectionist network, consisting of input units, output units, and intermediate or "hidden" units between input and output. The network has one special characteristic: Units at the hidden layers are replicated in many banks of units so that useful input patterns can be detected where ever they occur in the speech input (See Figure 5). The network is trained with a corpus of examples of human phonemes, and learns to recognize new examples with great accuracy. It does this, in spite of the fact that superficially, examples of different phoneme classes often seem more similar than examples from the same class.

--- INSERT FIG 5 ABOUT HERE ---

Researchers who have attempted to hand-craft detectors for properties of speech signals that signal identity-relevant properties have been hindered by the subtle trade-offs of cues and complex contingent relations between parts of the input that carry the important identity information. The connectionist learning rule solves this kind of problem by allowing the assignment of graded weights to graded, non-linear combinations of inputs. The gradual nature of the learning process means that those characteristics of the input that do the most work in discriminating the classes of inputs are learned first; gradually, with continued training, the subtler interdependencies are also acquired.

It is important to note in this and other applications that careful engineering is required to produce a system that works. The basic principles of Connectionist AI -- graded adjustment of graded connection strengths -- are supplemented in this system with several principles that are specific to the problem. The TDNN is explicitly designed to capture the shift-invariance of speech, for example, and this is largely responsible for its success.

Efficient programming of motor sequences. Motor control is an excellent example of a constraint satisfaction problem. In reaching for an object we must set the joint angles appropriately to achieve the target. In general, there

may be a very large number of solutions to this problem. Additional constraints come from efficiency criteria. If we wish to carry out a sequence of actions, reaching for a number of targets in succession, it would be desirable to minimize the motion of each joint as we move from one target to the next. But this constraint cannot be so strong as to prevent us from solving the initial problem of reaching the intended targets. A recent model by Jordan (1989) shows how connectionist learning schemes can discover solutions to motor-control problems of this type. Jordan's network adjusts connection strengths to reduce the error in positioning the tip of a moving articulator. The network also adjusts connection strengths to reduce the amount of movement of each joint angle in moving from one position in space to the next. Each constraint is graded; they work together to cause the articulator to follow a simple and near-minimal trajectory through joint-angle-space in moving from point to point on the x-y plane.

Language understanding. Language understanding is an excellent example of a constraint satisfaction problem. Though it is customary to assume that syntactic knowledge can be captured by a small number of rules, in fact it turns out that these rules generally exert graded influences on interpretation, and these influences can be overridden by semantic factors. St. John and McClelland (in press) have shown that connectionist networks can be trained to interpret sentences embodying these graded constraints in a way which converges toward a kind of probabilistic optimality, in the following sense. The outputs of the network are interpreted as probabilities associated with various interpretative decisions about the input sentence. Through learning, the network adjusts connection strengths to find a set of values which effectively encode the relative strengths of these different constraints.

It must be emphasized that in all of these cases it is highly unlikely that the required values of the graded constraints could be derived *a priori*. While there may be predispositions to perform certain kinds of transformations or to take certain types of constraints into account that are built into the processing

mechanisms, their ability to adjust connection strengths so as to find just the right balance of influences is crucial to the success of the models in all cases.

New Conceptions as Well as New Solutions

One of the themes of connectionist research is that it can change the way we think about certain kinds of issues in theories of intelligent information processing. There are many examples of this. Let us take the simple problem of learning something like the past tense of English. In the old way of thinking, the following basic assumptions were unquestioned:

- The knowledge that underlies our ability to form past tenses of novel words is stored in the form of rules.
- Knowledge of how to form the past tense of exceptions is stored in a mental "lexicon" or word-list.
- Acquisition of the past tense amounts to formulating, testing, and refining, rules and adding exceptions to the list.

Now we see that different assumptions are possible:

- The knowledge that underlies performance on novel items may be in a set of connections;
- The knowledge that allows us to deal with exceptions may be in the same set of connections;
- The acquisition process may simply be a process of connection strength adjustment.

As an illustration of this theme, we now consider how connectionist models may begin to suggest an elaboration of the classical types of automata enumerated by Chomsky. Specifically, we will describe a model in which the classical finite-state automaton is a special case of a broader class of what might be called "Graded State Machines". These machines capture some of the characteristics that Chomsky noted were lacking in finite state machines, and which caused him to reject such machines as candidates for representing human knowledge of natural language. In this case, we see that connectionist models hold out the hope of increasing our taxonomy of basic mechanism

types, and possibly of changing some of the conclusions that we reach about what type of mechanism is required to perform a certain type of processing.[2]

--- INSERT FIG 6 ABOUT HERE ---

The network. In this research we used a network introduced by Elman (1988). This network has the potential to master an infinite corpus of sequences with the limited means of a learning procedure that is completely local in time. We call this network a "Simple recurrent network" or SRN. In the SRN (Figure 6) the pattern of activation on the hidden units at time step t-1, together with the new input pattern, is allowed to influence the pattern of activation at time step t. This is achieved by copying the pattern of activation on the hidden layer at time step t-1 to a set of input units -- called the 'context units' -- at time step t. All the forward connections in the network are subject to training via back-propagation.

First, we show that the SRN can learn to mimic closely a finite state automaton (FSA), both in its behavior and in its state representations. In particular, we show that it can learn to process an infinite corpus of strings based on experience with a finite set of training exemplars. We then show how the architecture can exploit the fact that it uses graded information to do things that FSA's cannot do.

--- INSERT FIG 7 ABOUT HERE ---

Discovering a finite-state grammar. In our first experiment, we asked whether the network could learn the contingencies implied by a small finite state grammar (Figure 7). The network was presented with strings derived from

---

[2]The text and figures used in this portion of this paper is adapted from Cleeremans, Servan-Schreiber, and McClelland, 1989, with Permission [Permission needed].

this grammar, and required to try to predict the next letter at every step. These predictions are context dependent since each letter appears twice in the grammar and is followed in each case by different successors.

A single unit on the input layer represented a given letter (six input units in total; five for the letters and one for a begin symbol 'B'). Similar local representations were used on the output layer (with the 'begin' symbol being replaced by an end symbol 'E'). There were three hidden units.

On each of 60,000 training trials, a string was generated from the grammar, starting with the 'B'. Successive arcs were selected randomly from the two possible continuations with a probability of 0.5. Each letter was then presented sequentially to the network. The activations of the context units were reset to 0.5 at the beginning of each string. After each letter, the error between the network's prediction and the actual successor specified by the string was computed and back-propagated. The 60,000 randomly generated strings ranged from 3 to 30 letters.

Three tests were conducted. First, we examined the network's predictions on a set of 70,000 random strings. During this test, the network is first presented with the 'B', and one of the five letters or 'E' is then selected at random as a successor. If that letter is predicted by the network as a legal successor (i.e, activation is above 0.3 for the corresponding unit), it is then presented to the input layer on the next time step, and another letter is drawn at random as its successor. This procedure is repeated as long as each letter is predicted as a legal successor until 'E' is selected as the next letter. The procedure is interrupted as soon as the actual successor generated by the random procedure is not predicted by the network, and the string of letters is then considered 'rejected'. A string is considered 'accepted' if all its letters have been predicted as possible continuations up to 'E'. Of the 70,000 random strings, 0.3 % happened to be grammatical, and 99.7 % were ungrammatical. The network performed flawlessly, accepting all the grammatical strings and

rejecting all the others.

In a second test, we presented the network with 20,000 strings generated at random from the grammar, i.e, all these strings were grammatical. Using the same criterion as above, all of these strings were correctly 'accepted'.

Finally, we constructed a set of very long grammatical strings -- more than 100 letters long -- and verified that at each step the network correctly predicted all the possible successors (activations above 0.3) and none of the other letters in the grammar.

--- INSERT FIG 8 ABOUT HERE ---

Analysis of internal representations. What kind of internal representations have developed over the set of hidden units that allow the network to associate the proper predictions to intrinsically ambiguous letters? We recorded the hidden units' activation patterns generated in response to the presentation of individual letters in different contexts. These activation vectors were then used as input to a cluster analysis program, that groups them according to their similarity. Figure 8 shows the results of such an analysis conducted on a small random set of grammatical strings. The patterns of activation are grouped according to the nodes of the grammar: all the patterns that are used to predict the successors of a given node are grouped tog independently of the current letter. This observation sheds some light on the behavior of the network: at each point in a sequence, the pattern of activation stored over the context units provides information about the current node in the grammar. Together with information about the current letter (represented on the input layer), this contextual information is used to produce a new pattern of activation over the hidden layer, that uniquely specifies the next node. In that sense, the network closely approximates the FSA that would encode the grammar from which the training exemplars were derived. However, a closer look at the cluster analysis reveals that within a cluster corresponding

to a particular node, patterns are further divided according to the path traversed before the node is reached. For example, looking at the bottom cluster -- node #5 -- patterns produced by a 'VV', 'PS', 'XS' or 'SXS' ending are grouped separately by the analysis: they are more similar to each other than to other examples of paths leading to node #5. This tendency to preserve information about the path is not a characteristic of traditional finite-state automata.

Encoding path information. In the experiments described above, we used strings generated from a finite-state automaton. With these strings as inputs, we were able to train the network to imitate, very closely, the finite-state automaton that generated the training examples. But the network is not inherently a finite-state automaton of the usual kind; rather, it makes use of *Graded* state information. For this reason, we tend to call the SRN a *Graded State Machine*. In this section, we show how the SRN can make use of this graded state information to encode information about the path leading to a node in the grammar when it is relevant to predicting the identity of the next element of a string.

When strings generated from the small finite-state grammar may only have a maximum of eight letters, the prediction following the presentation of the same letter in position number six or seven may be different. For example, following the sequence 'TSSSXXV', 'V' is the seventh letter and only another 'V' would be a legal successor. In contrast, following the sequence 'TSSXXV', both 'V' and 'P' are legal successors. A network with 15 hidden units was trained on 21 of the 43 legal strings of length 3 to 8. It was able to use the small activation differences present over the context units - and due to the slightly different sequences presented - to master contingencies such as those illustrated above. Thus, after 'TSSXXV', the network predicted 'V' or 'P'; but after 'TSSSXXV', it only predicted 'V' as a legal successor.

Learning progresses through three phases. During the first phase, the

context information tends to be ignored because the patterns of activation on the hidden layer - of which the former are a copy - are changing continually as a result of the learning algorithm. In contrast, the network is able to pick up the stable association between each letter and all its possible successors. At the end of this phase, the network thus predicts all the successors of each letter in the grammar, independently of the arc to which each letter corresponds. In the second phase, patterns copied on the context layer now provide a unique code designating which letter preceded the current letter, and the network can exploit this stability of the context information to start distinguishing between different occurrences of the same letter -- different arcs in the grammar. Finally, in a third phase, small differences in the context information that reflect the occurrence of previous elements can be used to differentiate position-dependent predictions resulting from length constraints (see Servan-Schreiber, Cleeremans and McClelland, 1988, for more details).

This progression through three stages is, as a matter of fact, an entirely continuous process. It occurs gradually, as connection strengths are adjusted by the back propagation learning procedure. Depending on the inputs used in training, the network may end up mimicking a finite state automaton; alternatively, it may use the graded nature of the states of the network as the basis for making prediction relevant distinctions that reflect the prior elements in the string.

Processing embedded sequences. We have seen that the simple recurrent network is a traditional finite-state automaton only as a special case. Otherwise, it exploits the graded states of the network in ways which allow it to preserve information about early parts of a string until it is relevant later. This ability to make use of long-distance dependencies is, of course, an important part of human language processing abilities; a part that cannot be efficiently captured by traditional finite state machines.

--- INSERT FIG 9 ABOUT HERE ---

To test the SRN's ability to handle long-distance dependencies, we presented examples generated from the grammar shown in Figure 9. Although there are circumstances in which the network can fail to learn to be able to traverse the embedding, we have been able to show that in fact it can be trained to make correct predictions when exiting from the embedded grammar (although in most cases this ability degrades with the length of the embedding). For present purposes what is interesting is to discover what kind of a machine the simple recurrent network has become when it has learned successfully. It has some interesting characteristics which set it apart from classical finite state automata. A classic automaton, such as the one shown in Figure 9 itself, is only able to process the strings successfully by having two copies of the states between the initial letter in the string and the final letter. One copy is used after an initial P, the other is used after the initial T. This is inefficient, since the embedded material is the same in both cases. To capture this similarity in a simple and elegant way, it is necessary to use a more powerful machine, such as a recursive transition network. In this case, the embedding is treated as a subroutine which can be "called" from different places. A return from the call ensures that the grammar can correctly predict whether a T or P will follow. This ability to handle long-distance dependencies without duplication of the representation of intervening material lies at the heart of the arguments that have lead to the use of recursive formalisms to represent knowledge of language.

But the graded characteristics of the SRN allows the processing of embedded material as well as the material that comes after the embedding, without duplicating the representation of intervening material and without actually making a subroutine call. The states of the SRN can be used SIMULTANEOUSLY to indicate where the network is inside the embedding and to indicate the prior history of processing. The identity of the initial letter simply *shades* the representation of states inside the embedding, so that

corresponding nodes have similar representations, and are processed using overlapping portions of the knowledge encoded in the connection weights. Yet the shading that the initial letter provides allows the network to carry information about the early part of the string through the embedding, thereby allowing the network to exploit long-distance dependencies.

The ability to exploit long-distance dependencies is an inherent and basic aspect of human language processing capabilities, and it lies at the heart of the general belief that a recursive computational machine is necessary for processing natural language. The experiments we have done with SRN's suggest another possibility: It may be that long-distance dependencies can be processed by machines that are simpler than fully recursive machines, as long as they make use of graded state information. Of course, true natural language is far more complex that the simple strings that can be generated by the machine shown in Figure 9, so we cannot say for sure at this time whether Graded state machines will be able to process all aspects of natural language. However, our experiments indicate already that they are more powerful in interesting ways than traditional finite state automata. Certainly, the SRN should be seen as a new entry into the taxonomy of computational machines. Whether it (or some other example of the broader class of Graded State machines, of which the SRN is one of the simplest) will ultimately turn out to prove sufficient for natural language processing, remains to be established in further research.

Conclusion

This paper began with the idea that humans exploit graded information, and that computational mechanisms that aim to emulate the natural processing capabilities of humans should exploit this kind of information as well. Connectionist models do exploit graded information, and this gives them many of their attractive characteristics. It not only accounts for their ability to perform natural cognitive tasks, but also lies at the heart of their ability to

learn to perform these tasks. Through the gradual adjustment of connection strengths, networks can learn to emulate, and even surpass, the capabilities of conventional computational machines.

References

Cleeremans, A., Servan-Schreiber, D., & McClelland, J. L. (1989). Finite state automata and simple recurrent networks. *Neural Computation.*

Elman, J.L. (1988). Finding structure in time. CRL Technical report 9901. Center for Research in Language, University of California, San Diego, CA.

Jordan, M. I. (1989). Generic constraints on underspecified target trajectories. *Proceedings of the International Joint Conference on Neural Networks.* New York: IEEE Press.

Labov, W. (1973). The boundaries of words and their meanings. In C.-J. N. Bailey & R. W. Shuy (Eds.), *New ways of analyzing variation in English*, vol. 1. Washington: Georgetown University Press.

Lang, K. (1987). *Connectionist Speech Recognition.* Ph.D. thesis proposal, Carnegie Mellon University.

McClelland, J. L. (1985). Putting knowledge in its place: A scheme for programming parallel processing structures on the fly. *Cognitive Science, 9,* 113-146.

McClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception, Part I: An account of basic findings. *Psychological Review, 88,* 375-407.

McClelland, J.L. (1988). The case for interactionism in language processing. In: M. Coltheart, (Ed.), Attention and Performance XII, London: Erlbaum.

McClelland, J. L., & Rumelhart, D. E. (1985). Distributed memory and the representation of general and specific information. *Journal of Experimental Psychology: General, 114,* 159-188.

McClelland, J. L., Rumelhart, D. E., and Hinton, G. E. (1986). The appeal of parallel distributed processing. In Rumelhart, D. E., McClelland, J. L. and the PDP research group (Eds), *Parallel Distributed Processing: Explorations in the microstructure of cognition, Volume I.* Cambridge, MA: MIT Press.

Pinker, S. & Prince, A. (1988). On Language and Connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition,* **28**, 59-108.

Plunkett, K. & Marchman, V. (1989). Pattern association in a back propagation network: Implications for child language acquisition. CRL Techincal Report #8902, Center for Research in Language, C-008, University of California, Dan Diego, La Jolla, CA 92093.

Reber, A.S. (1967). Implicit learning of artificial grammars. Journal of Verbal

Learning and Verbal Behavior, 5, 855-863.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). Learning internal representations by backpropagating errors. *Nature*, **323**, 533-536.

Rumelhart, D. E., & McClelland, J. L. (1986). On learning the past tenses of English verbs. In J. L. McClelland, D. E. Rumelhart, & the PDP research group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Volume II.* Cambridge, MA: MIT Press.

Rumelhart, D. E., McClelland, J. L. & the PDP research group. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition. Volumes I & II. Cambridge, MA: MIT Press. [Also include reference to Japanese Edition!]*

Sejnowski, T.J. and C. Rosenberg. (1986. NETtalk: A parallel network that learns to read aloud. Technical Report JHU-EECS-86-01, Johns Hopkins University.

Servan-Schreiber, D., A. Cleeremans, and J.L. McClelland. (1988). Learning sequential structure in simple recurrent networks. Technical report CMU-CS-183, Carnegie Mellon University.

St. John, M., & McClelland, J. L. (in press). Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence.*

Waibel, A. (1989). Modular construction of time-delay neural networks for speech recognition. *Neural Computation, 1*, 39-46.

Figure Captions

Figure 1. Various objects used in studies of people's concept of a cup. From Labov (1973).

Figure 2. The interactive activation model of word perception of McClelland and Rumelhart (1981).

Figure 3. Processing of ambiguous inputs in the interactive activation model. Shadows are used to make portions of one of one letter ambiguous in each display. Activations of selected letter- and word- level units are shown for two displays containing ambiguous letters. On the left, the middle panel shows letter-level activations for letters in the fourth letter position; on the right, the middle panel shows activations of units for letters in the second letter position. From McClelland, Rumelhart, and Hinton (1986).

Figure 4. The model of past-tense learning from Rumelhart and McClelland (1986). A pattern representing the present tense of a word is presented at left; activation propagates from left to right through the network. The output represents the network's interpretation of the past tense of the word.

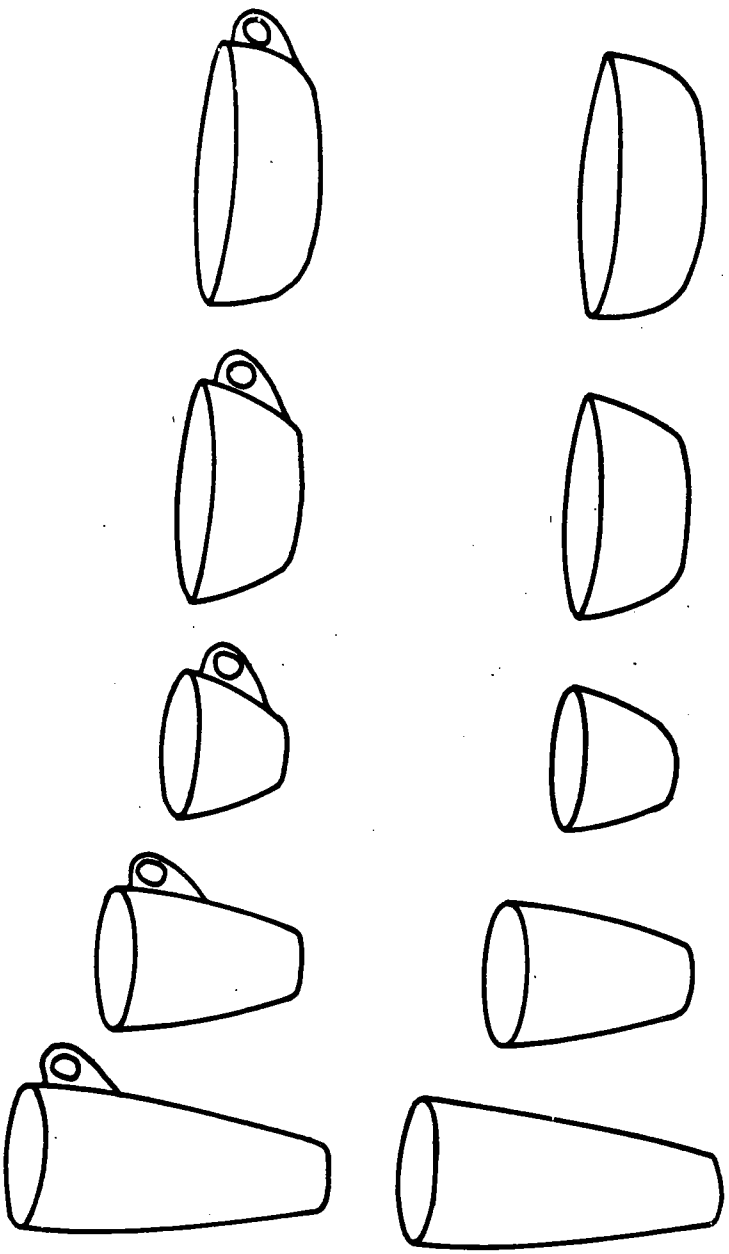
Figure 5. The Time Delay Neural Network (TDNN). At the bottom level, spectral coefficients representing a 150 msec portion of speech containing a token of the syllable BA is shown. These spectral coefficients are represented as activations of connectionist processing units. Units at upper levels receive inputs from subsets of units at lower levels. These subsets correspond to temporal windows of a small fixed size. The network is called a time-delay network because the input that each hidden unit sees can be thought of as representing the input at a particular time, together with delayed copies of the input from a few previous time steps. From Waibel (1989).

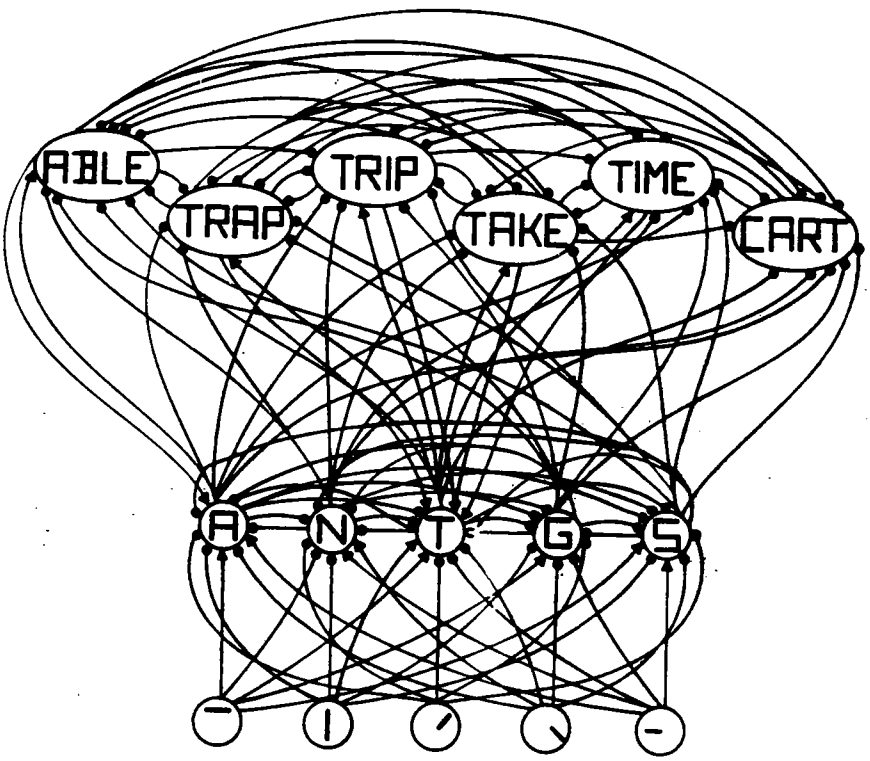
Figure 6. A simple recurrent network of the type introduced by Elman (1988) used in the experiments on learning artificial grammars. From Servan-Schreiber et al (1988).

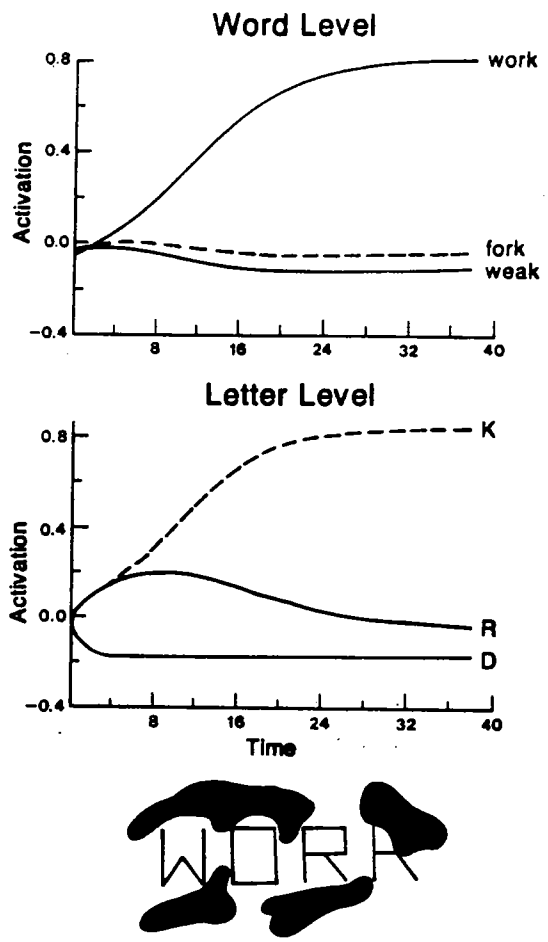
Figure 7. The simple finite state automaton used to generate strings that were presented to the FSA in Figure 6. Strings were generated by starting at the left with B, and proceeding to the first node, then successively choosing with equal probability among the arcs emanating from the current node, concatenating the letter on the arc onto the end of the string, and proceeding to the node at the end of the arc. From Servan-Schreiber et al (1988).

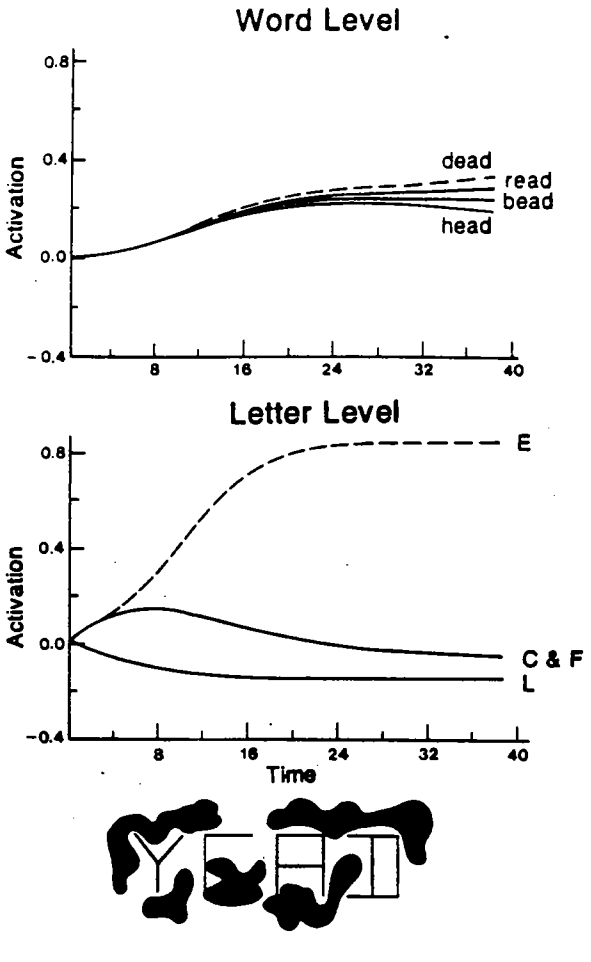
Figure 8. Hierarchical cluster analysis of states (patterns of activation on the hidden units) at different points in processing each of several randomly selected sequences. The diagram shows that the internal representations in this case represent the node in the grammar that has been reached, not the path that lead to the node. From Servan-Schreiber et al (1988).

Figure 9. A complex finite state automaton containing two identical embedded sub-automata. Adapted from Servan-Schreiber et al (1988).



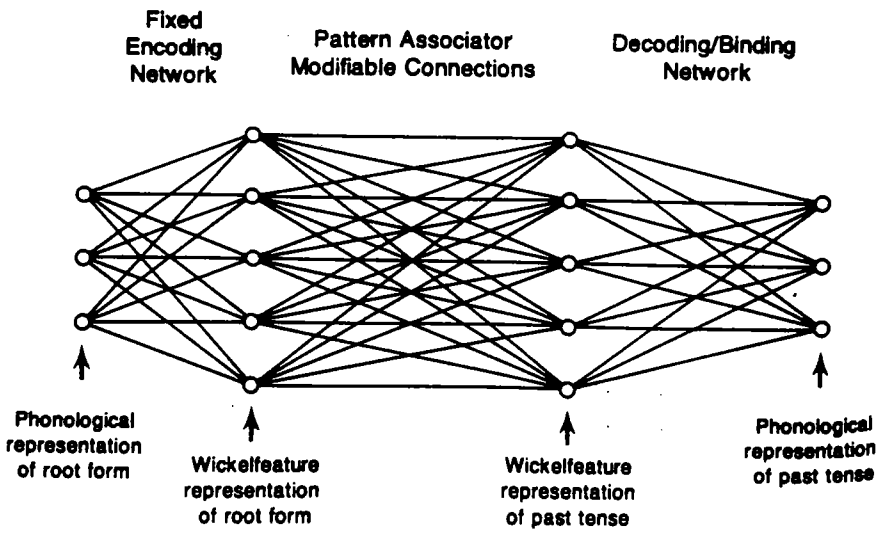

ABLE
TRIP
TIME
TRAP
TAKE
CART
A
N
T
G
S

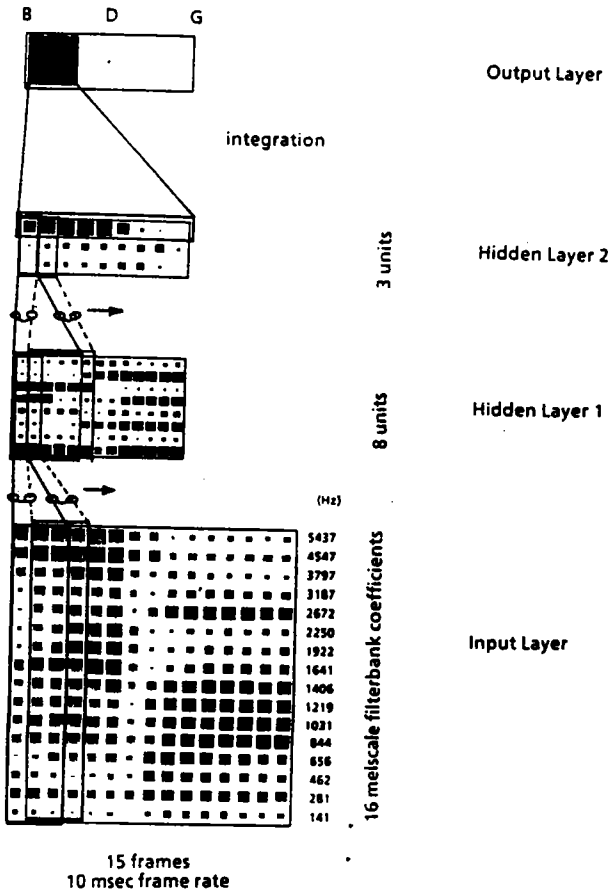

Word Level
Activation
0.8
0.4
0.0
-0.4
8
16
24
32
40
work
fork
weak
Letter Level
K
R
D
Time

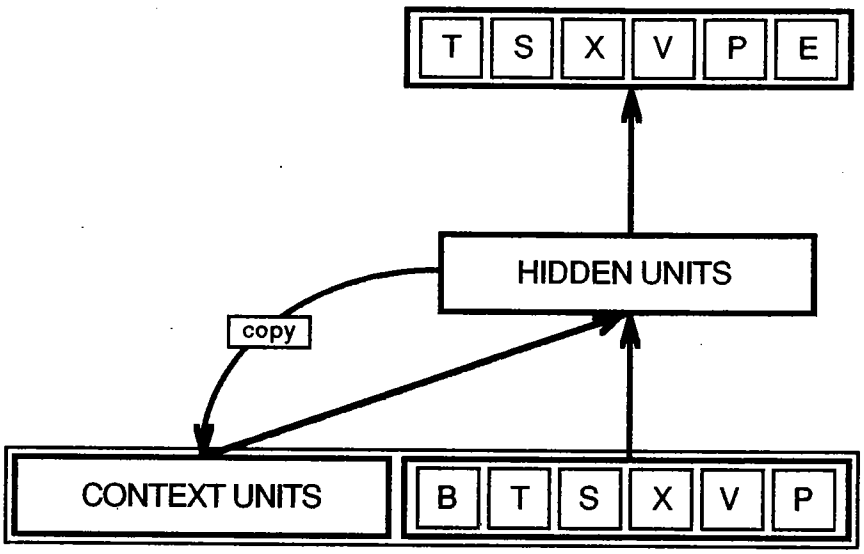

Word Level
Activation
0.8
0.4
0.0
-0.4
8
16
24
32
40
dead
read
bead
head
Letter Level
E
C & F
L
Time

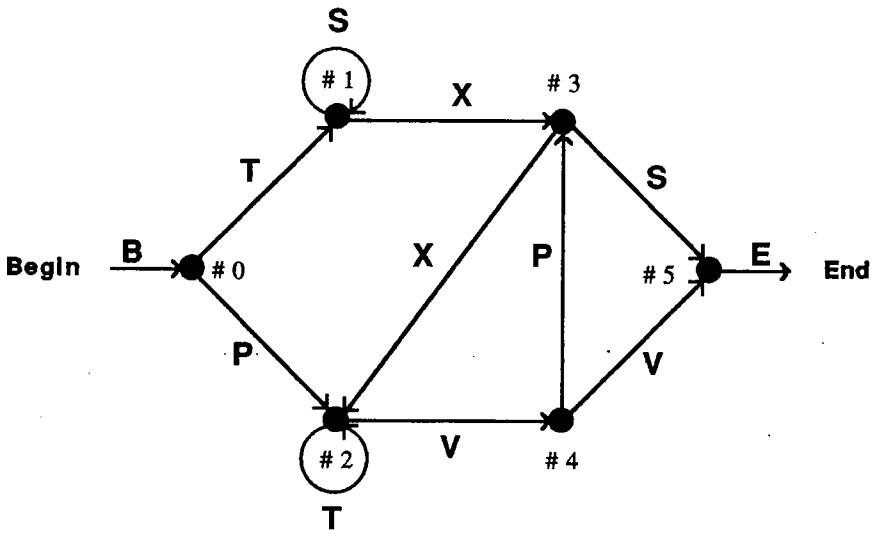
WORK

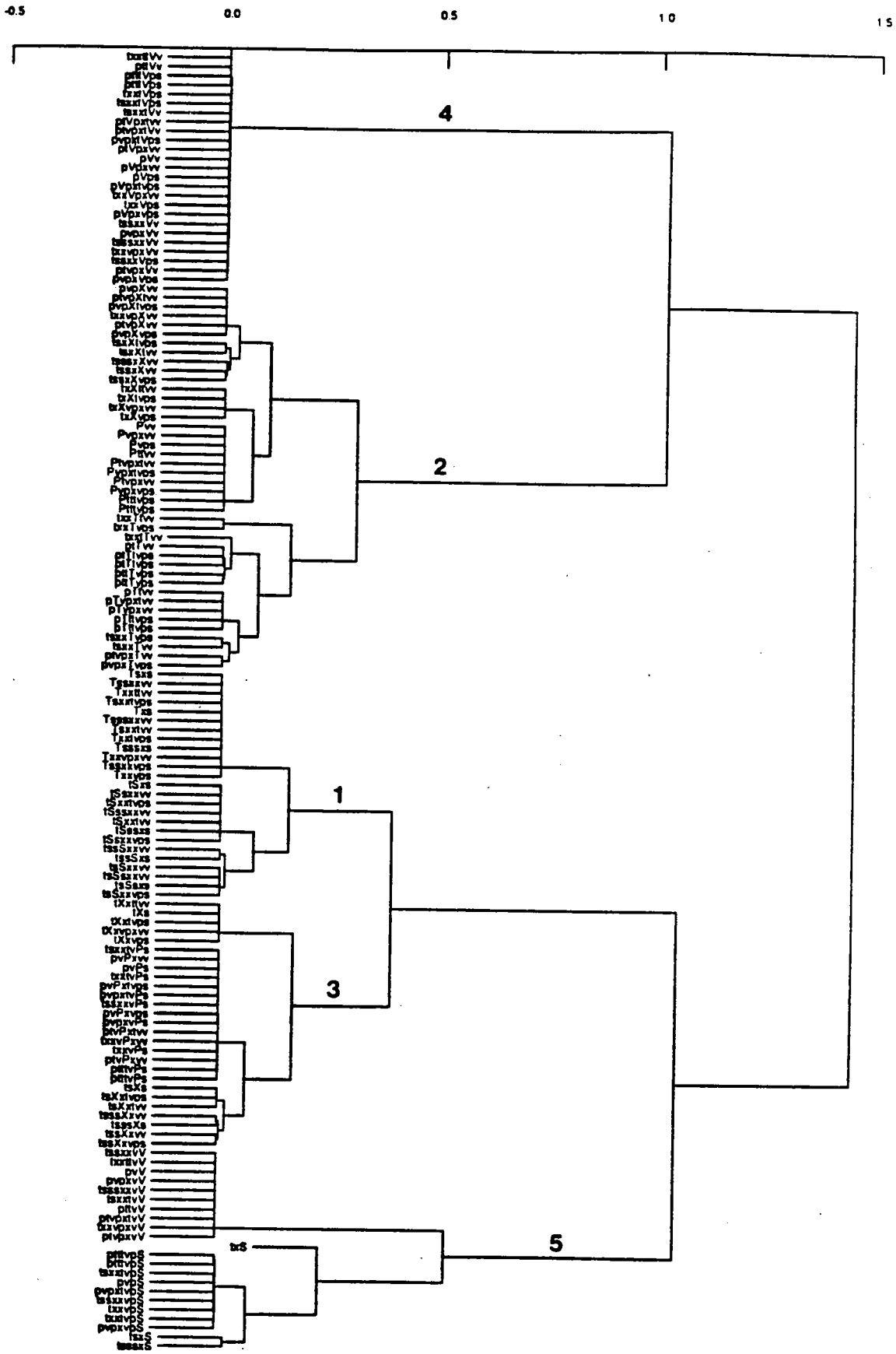
YEAR

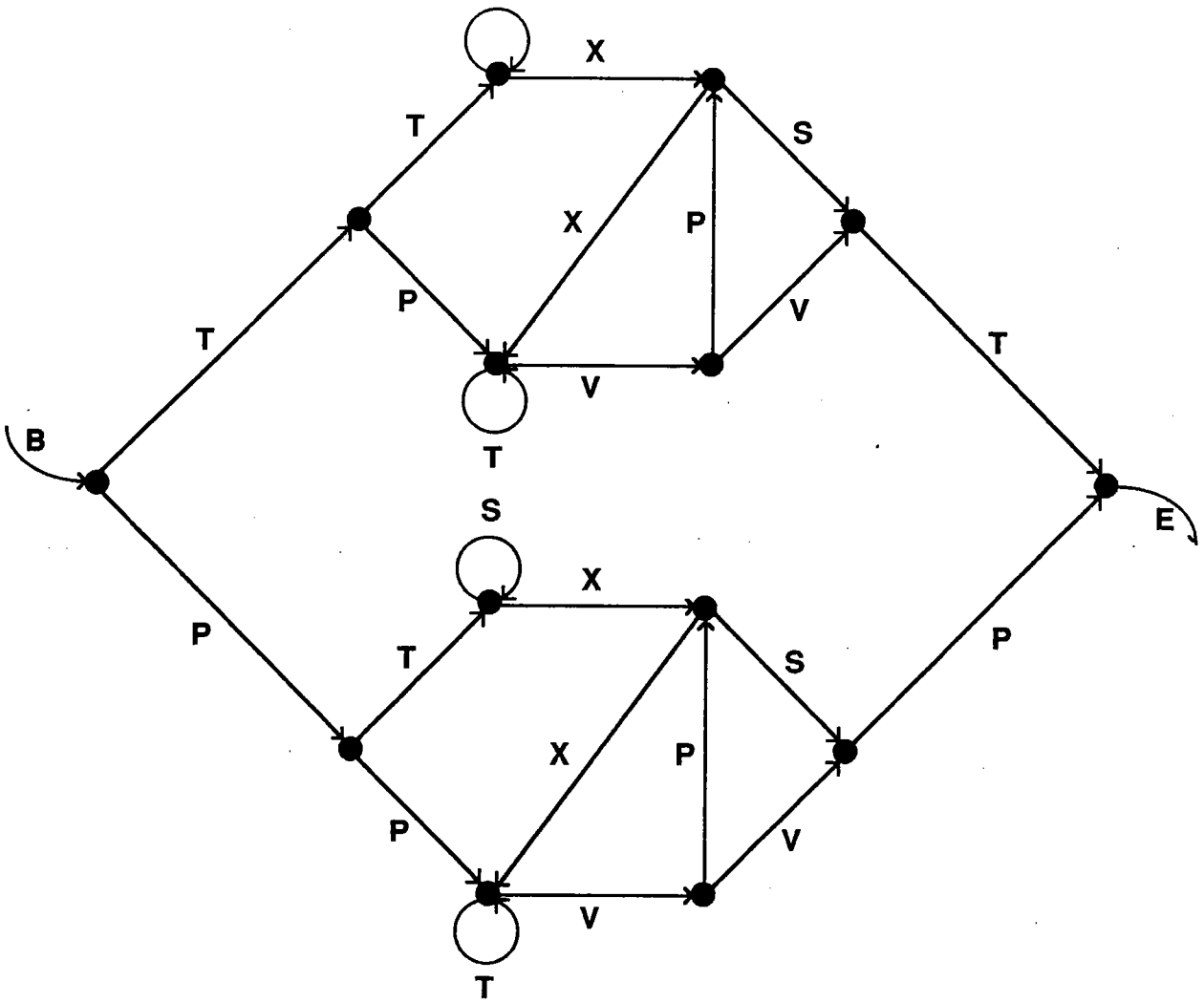

Fixed
Encoding
Network
Pattern Associator
Modifiable Connections
Decoding/Binding
Network
Phonological
representation
of root form
Wickelfeature
representation
of root form
Wickelfeature
representation
of past tense
Phonological
representation
of past tense


B
D
G
Output Layer
integration
3 units
Hidden Layer 2
8 units
Hidden Layer 1
(Hz)
5437
4547
3797
3187
2672
2250
1922
1641
1406
1219
1031
844
656
462
281
141
16 melscale filterbank coefficients
Input Layer
15 frames
10 msec frame rate


T
S
X
V
P
E
HIDDEN UNITS
copy
CONTEXT UNITS
B
T
S
X
V
P


S
1
X
3
T
S
Begin
B
0
X
P
5
E
End
P
V
V
2
4
T


-0.5
0.0
0.5
1.0
1.5
4
2
1
3
5


X
T
S
X
P
P
V
T
T
V
T
B
E
S
X
P
T
S
X
P
P
V
T
V
T