Communicated by Jeffrey Elman

Finite State Automata and Simple Recurrent Networks

Axel Cleeremans
Department of Psychology, Carnegie-Mellon University, Pittsburgh, PA 15213 USA

David Servan-Schreiber
Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213 USA

James L. McClelland
Department of Psychology, Carnegie-Mellon University, Pittsburgh, PA 15213 USA

We explore a network architecture introduced by Elman (1988) for predicting successive elements of a sequence. The network uses the pattern of activation over a set of hidden units from time-step $t-1$, together with element t, to predict element $t+1$. When the network is trained with strings from a particular finite-state grammar, it can learn to be a perfect finite-state recognizer for the grammar. When the network has a minimal number of hidden units, patterns on the hidden units come to correspond to the nodes of the grammar, although this correspondence is not necessary for the network to act as a perfect finite-state recognizer. We explore the conditions under which the network can carry information about distant sequential contingencies across intervening elements. Such information is maintained with relative ease if it is relevant at each intermediate step; it tends to be lost when intervening elements do not depend on it. At first glance this may suggest that such networks are not relevant to natural language, in which dependencies may span indefinite distances. However, embeddings in natural language are not completely independent of earlier information. The final simulation shows that long distance sequential contingencies can be encoded by the network even if only subtle statistical properties of embedded strings depend on the early information.

1 Introduction

Several connectionist architectures that are explicitly constrained to capture sequential information have been proposed. Examples are Time Delay Networks (for example, Sejnowski and Rosenberg 1986) — also called

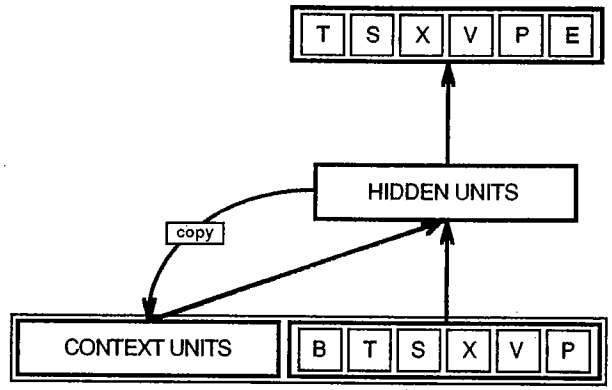

Figure 1: The simple recurrent network (Elman 1988). In the SRN, the pattern of activation on the hidden units at time step $t-1$, together with the new input pattern, is allowed to influence the pattern of activation at time step t. This is achieved by *copying* the pattern of activation on the hidden 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 backpropagation.

"moving window" paradigms — or algorithms such as backpropagation in time (Rumelhart *et al.* 1986; Williams and Zipser 1988). Such architectures use explicit representations of several consecutive events, if not of the entire history of past inputs. Recently, Elman (1988) has introduced a simple recurrent network (SRN) that has the potential to master an infinite corpus of sequences with the limited means of a learning procedure that is *completely local in time* (Fig. 1).

In this paper, 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 explore the capacity of this architecture to recognize and use nonlocal contingencies between elements of a sequence.

2 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 (Fig. 2). The network was presented with strings derived from 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.

2.1 Training. 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 backpropagated. The 60,000 randomly generated strings ranged from 3 to 30 letters (mean, 7; SD, 3.3).

2.2 Performance. 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

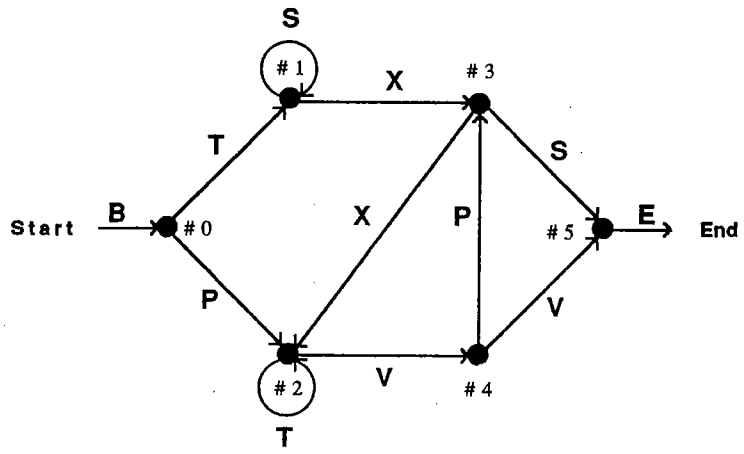

Figure 2: The small finite-state grammar (Reber 1967).

by the network as a legal successor (that is, 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*, that is, 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.

2.3 Analysis of Internal Representations. What kinds 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 3 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 together 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.

It must be noted that an SRN can perform the string acceptance tests described above and still fail to produce representations that clearly delineate the nodes of the grammar as in the case shown in Figure 3. This tendency to approximate the *behavior* but not the *representation* of the grammar is exhibited when there are more hidden units than are absolutely necessary to perform the task. Thus, using representations that correspond to the nodes in the FSA shown in figure 2 is only one way to act in accordance with the grammar encoded by that machine: representations correspond to the nodes in the FSA only when resources are severely constrained. In other cases, the network tends to preserve the separate identities of the arcs, and may preserve other information about the path leading to a node, even when this is not task relevant.

3 Encoding Path Information

In a different set of experiments, we asked whether the network could learn to use the information about the path that is encoded in the hidden units' patterns of activation. In one of these experiments, we tested whether the network could master length constraints. When strings generated from the small finite-state grammar may have a maximum of only 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 predicted only "V" as a legal successor.

How can information about the path be encoded in the hidden layer patterns of activation? As the initial papers about backpropagation pointed out, the hidden unit patterns of activation represent an "encoding" of the features of the input patterns that are relevant to the task. In the recurrent network, the hidden layer is presented with information about the current letter, but also — on the context layer — with an encoding of the relevant features of the previous letter. Thus, a given hidden layer pattern can come to encode information about the relevant features of two consecutive letters. When this pattern is fed back on the context layer, the new pattern of activation over the hidden units can come to encode information about three consecutive letters, and so on. In this

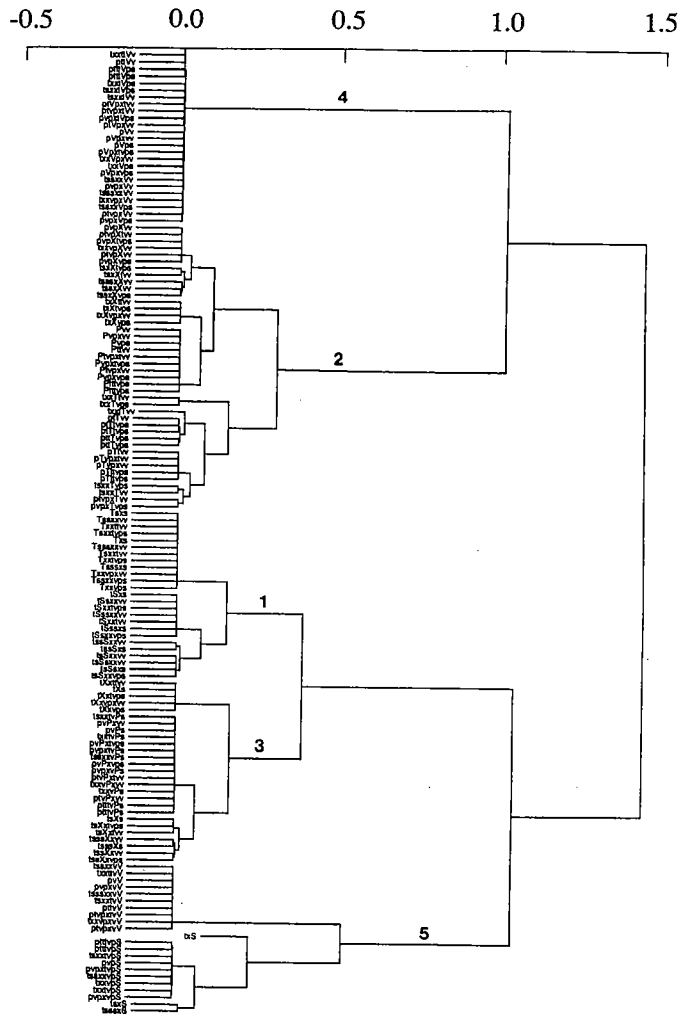

Figure 3: Hierarchical cluster analysis of the hidden unit activation patterns after 60,000 presentations of strings generated at random from the finite-state grammar. A small set of strings was used to test the network. The single uppercase letter in each string shown in the figure corresponds to the letter actually presented to the network on that trial.

manner, the context layer patterns can allow the network to maintain prediction-relevant features of an entire sequence.

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 are now represented by 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 *et al.* 1988, for more details).

It is important to note that information about the path that is not *locally relevant* tends not to be encoded in the next hidden layer pattern. It may then be lost for subsequent processing. This tendency is decreased when the network has extra degrees of freedom (that is, more hidden units) so as to allow small and locally useless differences to survive for several processing steps.

4 Processing Embedded Sequences

This observation raises an important question about the relevance of this simple architecture to natural language processing. Any natural language processing system must have the ability to preserve information about long-distance contingencies in order to correctly process sentences containing embeddings. For example, in the following problem of number agreement, information about the head of the sentence has to be preserved in order to make a correct grammatical judgment about the verb following the embedding:

The dog [that chased the cat] is playful
The dogs [that chased the cat] are playful

At first glance, however, information about the head of the sentence does not seem to be relevant for processing of the embedding itself. Yet, from the perspective of a system that is continuously generating expectations about possible succeeding events, information about the head *is* relevant within the embedding. For example, "itself" may follow the action verb only in the first sentence, whereas "each other" is acceptable only in the second. There is ample empirical evidence to support

the claim that human subjects do generate expectations continuously in the course of natural language processing (see McClelland 1988, for a review).

To show that such expectations may contribute to maintaining information about a nonlocal context, we devised a new finite-state grammar in which the identity of the last letter depends on the identity of the first one (see Fig. 4). In a first experiment, contingencies within the embedded subgrammars were identical (0.5 on all arcs). Therefore, information about the initial letter is *not* locally relevant to predicting the successor of any letter in the embedding.

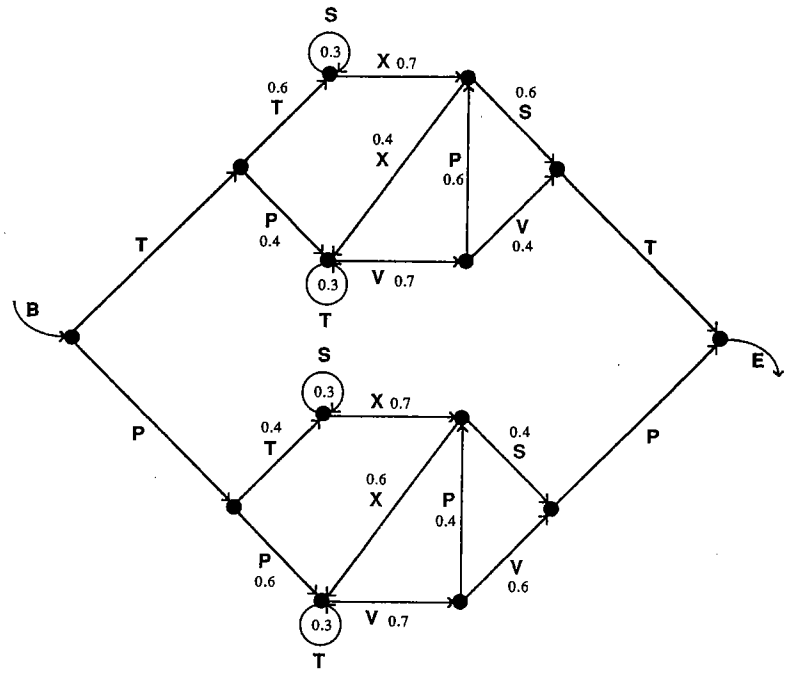

Figure 4: A complex finite-state grammar involving embedded sequences. The last letter is contingent on the first, and the intermediate structure is shared by the two branches of the grammar. In the first experiment, transition probabilities of all arcs were equal and set to 0.5. In the second experiment, the probabilities were biased toward the top arcs for the top embedding, and toward the bottom arcs for the bottom embedding. The numbers above each arc in the figure indicate the transition probabilities in the biased version of the grammar.

An SRN with 15 hidden units was trained on exemplars (mean length, 7; SD, 2.6) generated from this grammar for 2.4 million letter presentations. Even after such extensive training, the network indifferently predicted[1] both possible final letters, independently of the initial letter of the string.

However, when the contingencies within the embedded grammars depend, even very slightly, on the first letter of the string, the performance of the network can improve dramatically. For example, we adjusted the transition probabilities in the subgrammars such that the top subgrammar was biased toward the top arcs and the bottom subgrammar was biased toward the bottom arcs (see Fig. 4). An SRN trained on exemplars (mean length, 7; SD, 2.6) derived from this biased grammar performed considerably better. After 2.4 million presentations, the network correctly predicted the last letter of the string in 66.9% of the trials. There were 11.3% errors — that is, cases where the incorrect alternative was predicted — and 21.8% cases in which both alternatives were predicted about equally. Accuracy was very high for strings with a short embedding, and fell gradually to chance level at seven embedded elements. Note that we tested the network with strings generated from the *unbiased* version of the grammar. The network must therefore have learned to preserve information about the first letter throughout the embedding independently of whether the particular embedding is probable or not given the first letter. Moreover, with the biased grammar, some embeddings have exactly the same probability of occurring during training independently of whether the first letter is a "T" or a "P" (e.g., "PVPXVV"). As we have seen, training with such exemplars makes it very difficult[2] for the network to maintain information about the first letter. Yet, when such embeddings are part of an ensemble of embeddings that *as a whole* has a probability structure that depends on the first letter, information about this first letter can be preserved much more easily.

5 Conclusion

We have presented a network architecture first explored by Elman (1988) that is capable of mastering an infinite corpus of strings generated from a finite-state grammar after training on a finite set of exemplars with a learning algorithm that is local in time. When it contains only a minimal set of hidden units, the network can develop internal representations that correspond to the nodes of the grammar, and closely approximates the corresponding minimal finite-state recognizer. We have also shown that the simple recurrent network is able to encode information about long-distance contingencies as long as information about critical past events is relevant at each time step for generating predictions about potential alternatives. Finally, we demonstrated that this architecture can be used to maintain information across embedded sequences, even when the probability structure of ensembles of embedded sequences depends only subtly on the head of the sequence. It appears that the SRN possesses characteristics that recommend it for further study as one useful class of idealizations of the mechanism of natural language processing.

[1] We considered that the final letter was correctly or mistakenly predicted if the Luce ratio of the activation of the corresponding unit to the sum of all activations on the output layer was above 0.6. Any Luce ratio below 0.6 was considered as a failure to respond (that is, a miss).

[2] Other experiments showed that when the embedding is *totally independent* of the head, the network's ability to maintain information about a nonlocal context depends on the number of hidden units. For a given number of hidden units, learning time increases exponentially with the length of the embedding.

Acknowledgments

Axel Cleeremans was supported in part by a fellowship from the Belgian American Educational Foundation, and in part by a grant from the National Fund for Scientific Research (Belgium). David Servan-Schreiber was supported by an NIMH Individual Fellow Award MH-09696-01. James L. McClelland was supported by an NIMH Research Scientist Career Development Award MH-00385. Support for computational resources was provided by NSF(BNS-86-09729) and ONR(N00014-86-G-0146)

References

Elman, J.L. 1988. *Finding Structure in Time.* CRL Tech. Rep. 9901. Center for Research in Language, University of California, San Diego, CA.

McClelland, J.L. 1988. The case for interactionism in language processing. In *Attention and Performance* XII, M. Coltheart, ed. Erlbaum, London.

Reber, A.S. 1967. Implicit learning of artificial grammars. *J. Verbal Learning Verbal Behav.* **5**, 855–863.

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

Sejnowski, T.J., and Rosenberg, C. 1986. *NETtalk: A Parallel Network That Learns to Read Aloud.* Tech. Rep. JHU-EECS-86-01, Johns Hopkins University.

Servan-Schreiber, D., Cleeremans, A., and McClelland, J.L. 1988. *Learning Sequential Structure in Simple Recurrent Networks.* Tech. Rep. CMU-CS-183, Carnegie-Mellon University.

Williams, R.J., and Zipser, D. 1988. *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks.* ICS Tech. Rep. 8805. Institute for Cognitive Science, University of California, San Diego, CA.

Received 6 March 1989; accepted 20 April 1989.