Finding Linguistic Structure with Recurrent Neural Networks

Nick Chater
University of Edinburgh
Departments of Psychology &
7, George Square
Edinburgh EH8 9JZ
nicholas@uk.ac.ed.cogsci

Peter Conkey
University of Edinburgh
Artificial Intelligence
80 South Bridge
Edinburgh EH1 1HN
pbc@uk.ac.stir.cs

Abstract

Simple recurrent networks have been used extensively in modelling of learning various aspects of linguistic structure. We discuss how such networks can be trained, and empirically compare two training algorithms, Elman's "copyback" regime and back-propagation through time, on simple tasks. Although these studies reveal that the copyback architecture has only a limited ability to pay attention to past input, other work has shown that this scheme can learn interesting linguistic structure in small grammars. In particular, the hidden unit activations cluster together to reveal linguistically interesting categories. We explore various ways in which this clustering of hidden units can be performed, and find that a wide variety of different measures produce similar results and appear to be implicit in the statisticsof the sequences learnt. This perspective suggests a number of avenues for further research.

Introduction

Simple recurrent neural networks (SRNs) developed by Jordan (1986) and Elman (1988) provide a powerful tool with which to model the learning of many aspects of linguistic structure (for example, Elman 1990, 1991; Shillcock, Levy & Chater 1991) and there has been some exploration of their computational properties (Chater 1989; Cleermans, Servan-Schrieber & McClelland 1989; Servan-Schrieber, Cleeremans & McClelland 1991). The presence of recurrent connections allows past activation to influence current output, which means that output can respond sequential structure in the input. The extent to which such networks can be taught to learn interesting sequential structure depends on the learning algorithm employed. A natural approach is to apply the back-propagation training algorithm which has proved so successful in training non-recurrent feedforward networks to learn interesting static input-output patterns.

The structure of the paper is as follows. First we discuss a number of ways in which the back-propagation algorithm can be adapted to train recurrent networks to learn sequences, concentrating on two options, Elman's (1990) "copyback" scheme, and back-propagation through time (Rumelhart, Hinton & Williams 1986). We note that there are theoretical reasons to suppose that the copy-back regime will learn less well, and this conclusion is borne out in our simulations. However, the copy-back approach is computationally inexpensive and has provided impressive results in a number of language processing tasks. We investigate the scope of this method further, and follow Elman in investigating the nature of the hidden unit representations developed for a network which learns to predict the next element in sequences generated by a simple grammar. A number of very different measures over the hidden units are found to generate very similar syntactic/semantic clustering, and these clusters are also implicit in the statistics of the sequences learnt. This suggests that network performance can usefully be analysed in terms of the statistical structure of the input sequences, and that the applicability of SRNs to real natural language data can be assessed by analysing relevant aspects of its statistical structure

Training SRNs

Backpropagation cannot directly be applied to SRNs since the algorithm applies only to feedforward networks. For these, back-propagation performs gradient descent in the sum-squared error over the finite number of input-output pairs in the training set. To apply back-propagation to recurrent networks, the SRN must be "unfolded" into a feedforward network which is then trained in the conventional way. The most popular method of training SRNs involves unfolding the network by providing an additional input - the context units - which corresponds to the previous values of the hidden units (Elman 1990) (Figure 1a, 1c). The context units are dependent on the previous inputs, among which is the previous value of the context units. Hence the behaviour of the network is influenced not just by the current input but by the sequence of past inputs. While activation is propagated forwards through the network from arbitrarily far back in time, error is only propagated back to the context units.

An alternative approach is to unfold the network through several time steps (Rumelhart, Hinton & Williams 1986) so that each weight has several "virtual incarnations" and to back-propagate through the resulting network (Figure 1a, 1c). The overall weight change is simply the sum of the changes recommended for each incarnation. This "back-propagation through time" can in principle be back-propagated through the entire training history of the network (Rohwer personal communication) but is typically implemented by unfolding through a small number (here we shall use 5) time steps. The copy-back scheme can be viewed as a special case of back-propagation through time, in which the back-propagation of error stops at the first copy of the hidden units - the context units.

The more the network is unfolded, the better the approximation of the feedforward network to the underlying recurrent network, and the better the network learns to respond to sequential material. The minimal unfolding embodied in copy-back scheme should therefore produce the poorest learning, although it has the considerable advantage of being the cheapest computationally. This is borne out below in the comparitive studies below.

A natural assumption would be that the number of steps back that error is propagated will precisely fix the number of steps "back" the network can learn about. If this were true, the copy-back scheme would only be able to respond to the current and previous input, and would thus not be able to learn any interesting sequences. However, as long as the relevant temporally distant information "percolates through", even in some degraded form, to a point in the unfolded network to which error is propagated, the weights forward of that point can be adjusted to utilize that information successfully. Hence, the last point in the network to which error is propagated forms a "bottleneck", at which temporally more distant information must pass if the network is to be able to learn to respond to it (see Figure 2) (Chater 1989).

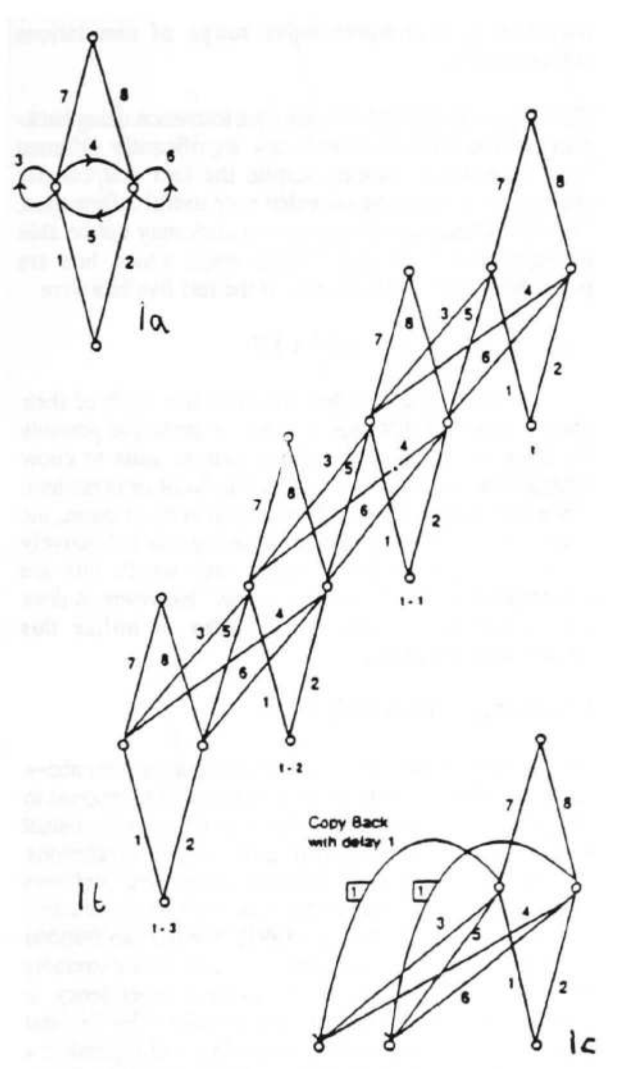

Figure 1 *Unfolding a recurrent neural network (1a), for back-propagation through time (1b) and copy-back training (1c).*

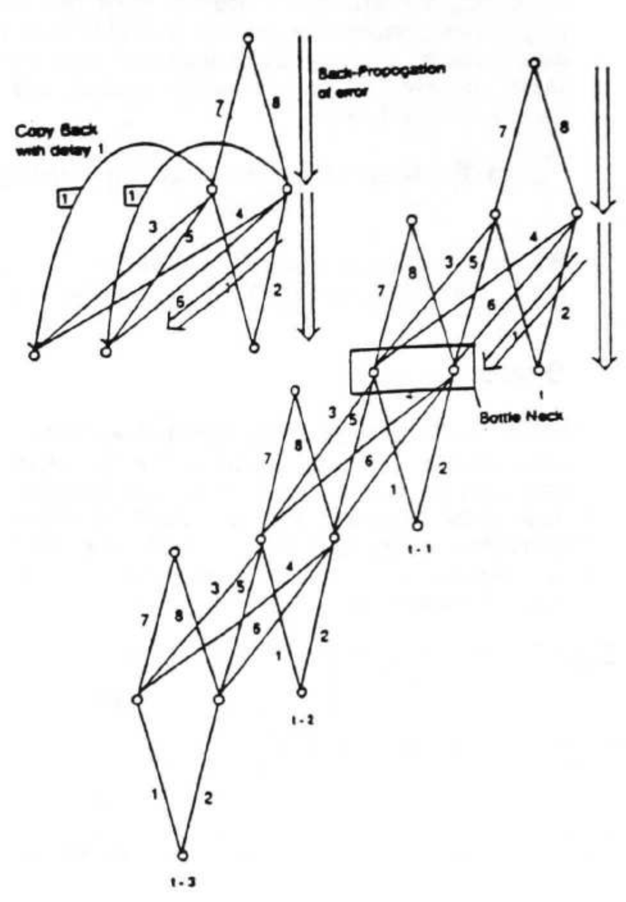

Figure 2 *Back-propagation in the copy-back training regime.*

In many tasks, temporally distant information beyond the bottleneck, is relevant to predicting intervening material. This means that the network encodes that information in its hidden units, to predict that more local information, which forces this information through the bottleneck. Hence the SRN is able to learn to respond to this distant information successfully. However, in a task in which temporally distant information is not correlated with intervening material, such as the task of learning to be delay line, reported below, learning with the copy-back scheme should be poor.

While our primary interest in this first set of simulations was comparing the performance of copy-back and back-propagation through time, a secondary interest was in the effect of using or not using context units in back-propagation through time. If the network is unfolded several time-steps (5 in the simulation we report), the contribution of the context units at the bottleneck to the final output may be very small, and the large number of intervening layers may make it difficult to learn to respond to this input, even if it is informative. From a theoretical point of view, not using context units is attractive, since the network can then be viewed as learning a fixed input-out put set (or a sample from a fixed distribution), and hence the proof that back-propagation performance gradient descent is valid. For most problems, the presence or absence of context units seems to have little effect on performance, and we discuss this briefly below.

Copy-Back and Backpropagation Through Time

We report simulations on two very simple tasks using binary sequences, discrete XOR and learning to the a delay line.

Discrete XOR

Consider a binary sequence in which two out of three bits are generated at random, and the third is the XOR of the previous two. The task is to attempt to predict the next value in the sequence. This task is difficult, since only every third bit can is in principle predictable. Optimal performance is to correctly predict these bits, and to output 0.5 otherwise.

Architecture	Hidden Units	Average squared errors		
		Posn 1	Posn 2	Posn3
copy back	4	0.278 ± 0.003	0.273 ± 0.003	0.16 ± 0.02
copy back	7	0.282 ± 0.002	0.288 ± 0.002	0.11 ± 0.01
copy back	10	0.283 ± 0.001	0.289 ± 0.002	0.10 ± 0.01
unfolded with contexts	2	0.267 ± 0.004	0.271 ± 0.002	0.20 ± 0.02
unfolded with contexts	3	0.276 ± 0.004	0.280 ± 0.003	0.16 ± 0.03
unfolded with contexts	4	0.275 ± 0.004	0.288 ± 0.004	0.18 ± 0.03
unfolded with contexts	5	0.278 ± 0.004	0.283 ± 0.003	0.12 ± 0.02
five unfolds no contexts	2	0.268 ± 0.004	0.270 ± 0.004	0.19 ± 0.02
five unfolds no contexts	3	0.282 ± 0.003	0.285 ± 0.003	0.12 ± 0.01
five unfolds no contexts	4	0.281 ± 0.003	0.286 ± 0.003	0.11 ± 0.02
five unfolds no contexts	5	0.289 ± 0.001	0.287 ± 0.001	0.089 ± 0.004

Table | : Performance on the discrete XOR task with 50 epochs of training

Copy-back and back-propagation schemes (both with and without context) were trained on XOR (Table 1). The results were averaged over 50 trials, with 50 training epochs over 3000 input-output pairs with learning rate 0.1 and momentum 0.9. For back-propagation through time, the net was unfolded 5 time steps. The weights were initialised randomly between -5 and 5. If the weight starts are smaller than this, "copyback" learning is slow, perhaps because for the copy-back regime, perhaps because for small inputs the sigmoid activation function is nearly linear, and hence unable to compute XOR . Notice that the standard deviations of the errors obtained are small throughout.

For a network of a given size, performance is far better with back-propagation through time than using the copy-back scheme, which require far more hidden units to attain comparable results. This pattern is consistently obtained in a comprehensive range of simulations (Conkey 1991).

Turning to our second concern, performance using back-propagation through time is not significantly different with or without context, despite the fact that context could in principle have provided very useful information. This is because the no context network may not be able to determine from just 5 time steps which bits are predictable and which are not. If the last five bits were

...0 1 1 1 0

then the third and fifth bits are both the XOR of their predecessors. In this case it is not in principle possible for these unfolded nets without context units to know whether the next bit is the result of an XOR or is random. Since this ambiguity occurs in almost 60% of cases, the ability to use past context to disambiguate (effectively storing a regular "pulse" indicating which bits are predictable) would be advantageous. However, it does not appear to be possible to learn to utilize this information in practice.

Learning to be a delay line.

The analysis of the copy-back learning algorithm above suggested that it should be poor at learning to respond to temporally distant input, unless the temporally distant information has been used in intermediate predictions. This suggests that while in many interesting problems (such as that of learning a grammar with some recursive structure, detailed in Elman (1991)) the net can respond to temporally distant information, this will extremely difficult if the nature of the distant dependency is independent of the intervening material. The simplest such task is learning to be a delay line - to reproduce a random binary input stream delayed by several time steps.

A recurrent network with n+1 hidden units can act as a delay line of n, given appropriate weights. One intuitively attractive solution is for the hidden units to act as buffers for the input so that one unit has output at time t of i(t-1), another i(t-2) and so on back to i(t-n). Figure 3 illustrates weights that would implement this solution for a delay line of 1 and a network with two hidden units.

Table 2 shows a typical sample of results. While back-propagation through time is able to learn the delay line task quite well (and with only n+1 hidden units for small delays n), the copy-back scheme can only learn to respond to small delays with relatively large numbers of hidden units. This is explicable in terms of the theoretical discussion above - the more hidden units the more likelihood that relevant information will by chance percolate through the network and thus that the network will be able to learn to use this information. Learning performance is also very much less consistent with the copy-back regime.

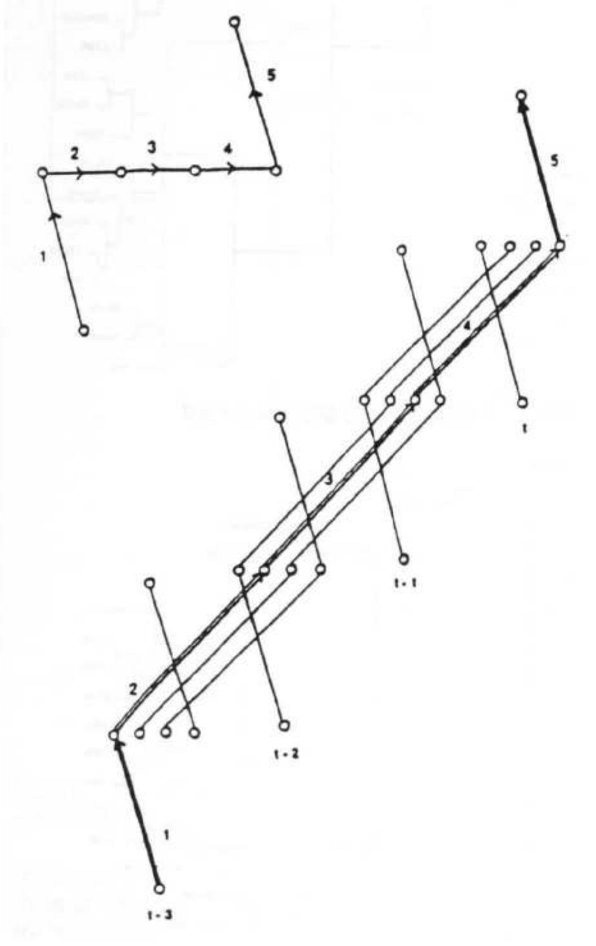

Figure 3 *Implementing a delay of n time steps with n+1 hidden units.*

The inability of the copy-back scheme to learn to respond to long time delays contrasts with good performance reported predicting dependencies in small scale language tasks where the intervening material is relevant (Elman 1991).

The results of these experiments bear out the theoretical analysis that back-propagation through time leads to better learning than the copy-back scheme. However, back-propagation through time is computationally more expensive, and the copy-back scheme may be able to learn many interesting tasks. One particularly intriguing result is that the averaged hidden unit patterns appear to encode the syntactic/semantic categories for a toy grammar (Elman; 1990). The studies reported below repeat, extend and analyse this result, and argue that such a clustering is to be expected given the statistics of the sequences leant.

Architecture	Delay	Hidden units	Learning rate	Average squared error	Passes
copy back	1	2	0.05	0.22	< 100
copy back	1	2	0.10	0.21	< 100
copy back	1	3	0.05	0.20	< 100
copy back	1	3	0.10	0.16	< 100
copy back	1	4	0.05	0.13	< 100
copy back	1	4	0.10	0.097	< 100
unfolded	1	2	0.10	0.001	3
copy back	2	3	0.05	0.254	< 100
copy back	2	3	0.10	0.256	< 100
copy back	2	4	0.05	0.262	< 100
copy back	2	4	0.10	0.260	< 100
copy back	2	7	0.05	0.278	< 100
copy back	2	7	0.10	0.264	< 100
copy back	2	10	0.05	0.160	< 100
copy back	2	10	0.10	0.239	< 100
unfolded	2	3	0.10	0.031	3

Table 2 : Learning to be a delay line

Incidently Recognising Linguistic Structure

Elman used to copy-back regime to train a net to predict the next item in a continuous text sequence, generated by a simple grammar (borrowed from Elman 1988; 1990). Whereas Elman represented each "word" by a random bit vector, used a completely localist representation, thus using 29 input units to represent the 29 words. As in Elman's simulations there is no explicit marker for the end of a sentence. 150 hidden units and 150 corresponding context units were used.

Conditional probabilities for the next word, given the sentence so far were calculated from the data set. The RMS errors relative to this benchmarkwere 0.2 per patternin both cases, whereas Elman obtained 0.05. This difference may be a result of our choice of a localist input representation. We then followed Elman in cluster analysing the hidden unit activation evoked on presentation of each word. These were averaged to give a single 150 element vector for each of the 29 words.

The results from a typical net (Figure 4) do not give as good a clustering as that obtained by Elman. There is poor separation of nouns and verbs, and some confusion between different classes of each. Clustering on the basis of current input may not be the best measure, since the hidden unit values must encode previous input relevant to prediction, not just the current word. We also clustered hidden unit states averaged by the entirey Hence a more attractive alternative is to average hidden unit patterns together on the basis of the word predicted.

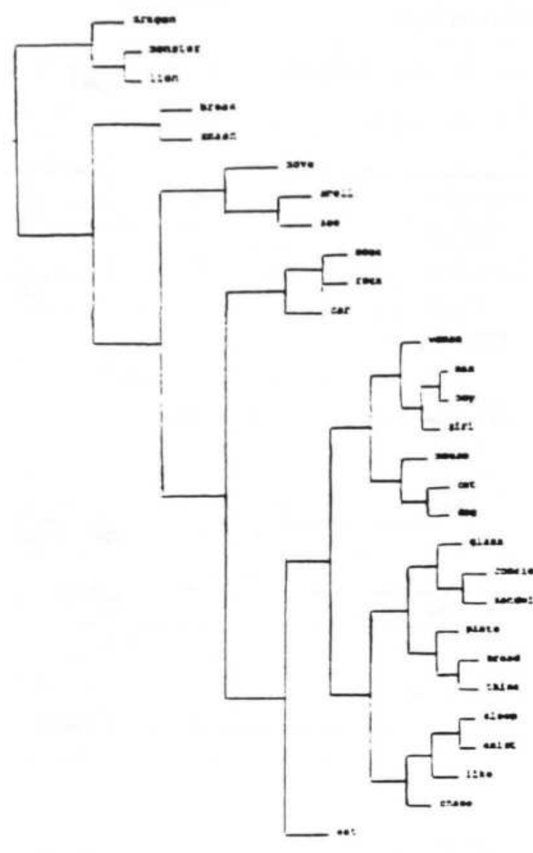

Figure 4 *Clustering by current word*

This measure, which completely cross-classifies the data with respect to the original measure, does indeed produce much better clusters, shown in Figure 5. Using this measure the clusters obtained well reflect the underlying syntactic categories of the grammar, with, for examples, nouns being separated from verbs, and different kinds of nouns being very well segrated and verbs segregated somewhat less precisely.

A further possibility is to cluster not the hidden unit pattern associated with an incoming word, but the change in hidden unit representation brought about by that word. Again a good clustering is obtained (Figure 6), comparable in quality with that obtained by clustering with respect to the target word.

It seems that a variety of measures of hidden unit values produce clusters corresponding to linguistically interesting categories. Is there an "optimal" clustering of this data set to which all of these measures are approximating? Not really - each of the measures considered above correspond to statistics of the data sets, which can be directly measured. For example, Elman's original measure of averaging hidden units on the basis

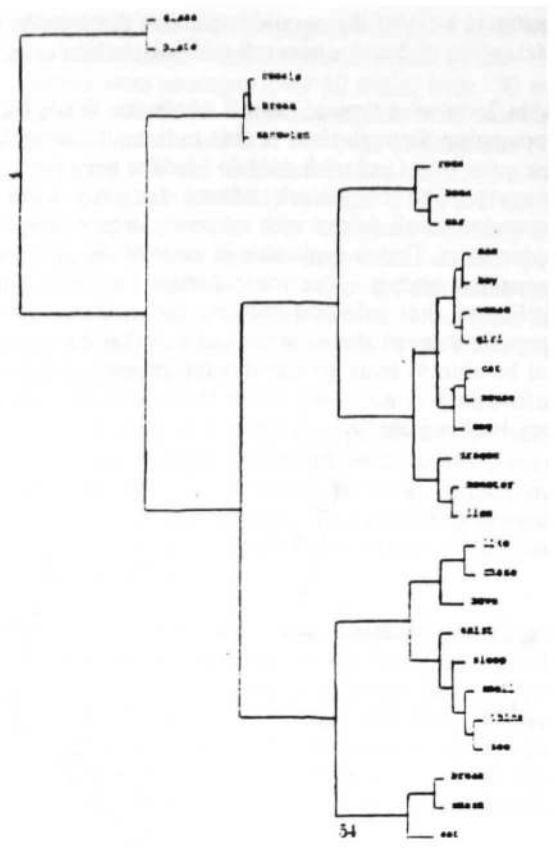

Figure 5 *Clustering by predicted word*

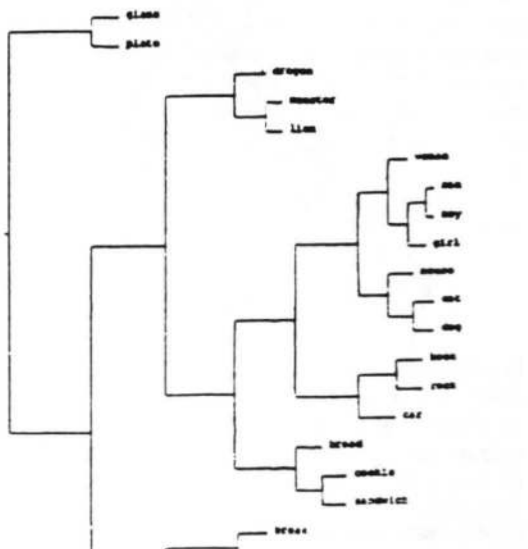

Figure 6 *Clustering by change of hidden unit pattern*

of the past word corresponds to grouping words by the conditional probabilities of successive words. We measured this quantity directly, and then cluster analysed (Figure 7) to produce very similar ressults to those obtained from the network (Figure 4) - this means that the network is successfully sampling the relevant statistic. Similar results can be obtained by comparing the two other measures with statistical analogues (clustering words on the basis of the conditional probabilities of the preceding words, and the change in conditional probabilities expected after a word is input,

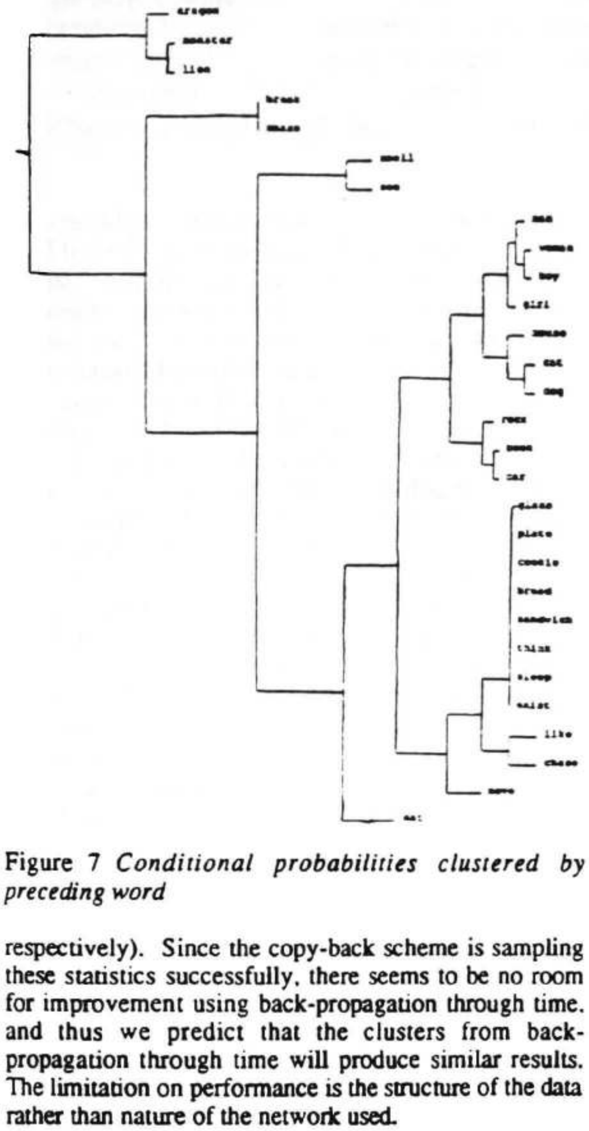

Figure 7 *Conditional probabilities clustered by preceding word*

respectively). Since the copy-back scheme is sampling these statistics successfully, there seems to be no room for improvement using back-propagation through time. and thus we predict that the clusters from back-propagation through time will produce similar results. The limitation on performance is the structure of the data rather than nature of the network used.

These results suggests that the hidden unit patterns that recurrent neural networks develop can be viewed as reflecting quite directly the statistical structure of the sequences learnt. Furthermore, particular statistical measures of hidden unit activation may closely correspond to a related statistic of the sequence itself.

Conclusion

The first set of simulations reported confirmed the theoretically motivated expectation that the back-propagation through time is superior to (less expensive) copy-back training for learning sequential structure. Experiments with large copy-back networks suggest that the hidden unit representation is successfully sampling statistics of the underlying sequential material, and we predict that back-propagation through time should, therefore, produce very similar clusters. Of course, if the underlying grammar, and hence the relevant statistics, are more complex, then back-propagation through time may then be able to sample these statistics better.

This suggests three interesting avenues for further research: 1) to investigate further the relationship between statistical analysis of the hiddenunit representations and direct analysis of the original data set, both using the copy-back and back-propagation through time regimes; 2) to explore real natural language data directly by cluster analysing using simple statistics to explore what peformance can be expected from a neural network model; 3) to investigate if statistics which are revealing of linguistic structure can be implemented more directly in a network, so that a full-size network can be built which is able to handle real natural language data. These last two avenues have recently been explored by Finch & Chater (1991) with encouraging results.

Chater, N. (1989) "Learning to Respond to Structures in Time" (Technical Report, Research Initiative in Pattern Recognition, St Andrews Road, Malvern, Worcs. RIPRREP/1000/62/89.

Cleeremans, A., Servans-Schreiber, D. & McClelland, J. L. (1989) Finite state automata and simple recurrent networks. Neural Computation, 1. 372-381.

Conkey, P. (1991) Sequence Prediction Using Re current Neural Networks. MSc Thesis, Department of Artificial Intelligence, University of Edinburgh.

Elman, J. L. (1988) Finding structure in time. Technical Report, CRL TR 8801, Centre for Research in Language, UCSD.

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

Elman, J. L. (1991) Distributed Representations, Simple Recurrent Neural Networks, and Grammatical Structure. Machine Learning, 7, 195-225.

Finch, S. & Chater, N. (1991) A Hybrid Approach to the Automatic Learning. *AISB Quarterly*, 78, 16-24.

Jordan, M. (1986) Serial order: a parallel distributed approach. Institute for Cognitive Science Report, 8604, University of California, San Diego.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986) Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds) *Parallel Distributed Processing* Vol 1, 318-362, Cambridge, Mass: MIT Press.

Servans-Schreiber, D., Cleeremans, A. & McClelland, J. L. (1991) Graded state machines: The representation of temporal contingencies in simple recurrent networks. *Machine Learning*., 7, 161-193.

Shillcock, R., Levy, J. & Chater, N. (1991) "A connectionist model of word recognition in continuous speech" Proceedings of the Cognitive Science Society of America Annual Conference, 340-345.