Knowledge Extraction and Recurrent Neural Networks: An Analysis of an Elman Network trained on a Natural Language Learning Task

Ingo Schellhammer*#, Joachim Diederich*, Michael Towsey*, Claudia Brugman**

* Neurocomputing Research Centre, Queensland University of Technology, QLD, 4001, Australia
# Dept of Information Systems, University of Muenster, D-48149 Muenster, Germany
**School of Languages, University of Otago, New Zealand
joachim@fit.qut.edu.au

Abstract

We present results of experiments with Elman recurrent neural networks (Elman, 1990) trained on a natural language processing task. The task was to learn sequences of word categories in a text derived from a primary school reader. The grammar induced by the network was made explicit by cluster analysis which revealed both the representations formed during learning and enabled the construction of state-transition diagrams representing the grammar. A network initialised with weights based on a prior knowledge of the text's statistics, learned slightly faster than the original network.

1. Introduction

Since their renaissance in the mid-1980s, Artificial Neural Network (ANN) techniques have been successfully applied across a broad spectrum of problem domains such as pattern recognition and function approximation. However despite these capabilities, to an end user an ANN is an arcane web of interconnected input, hidden, and output units. Moreover an ANN solution manifests itself entirely as sets of numbers in the form of activation function parameters and weight vectors. As such a trained ANN offers little or no insight into the process by which it has arrived at a given result nor, in general, the totality of "knowledge" actually embedded therein. This lack of a capacity to provide a "human comprehensible" explanation is seen as a clear impediment to a more widespread acceptance of ANNs.

In order to redress this situation, recently considerable effort has been directed towards providing ANNs with the requisite explanation capability. In particular a number of mechanisms, procedures, and techniques have been proposed and developed to extract the knowledge embedded in a trained ANN as a set of symbolic rules which in effect mimic the behaviour of the ANN. A recent survey conducted by Andrews et al. (1995) offered an insight into the modus operandi of a broad cross-section of such techniques.

In this paper we focus on the extraction of grammatical rules from trained Artificial Neural Networks and, in particular, Elman-type recurrent networks (Elman, 1990). Unlike Giles & Omlin (1993 a,b) who used an ANN to simulate a deterministic Finite State Automaton (FSA) representing a regular grammar, we have extracted FSA's from a network trained on a natural language corpus. The output of *k-means* cluster analysis is converted to state-transition diagrams which represent the grammar learned by the network. We analyse the prediction and generalisation performance of the grammar.

2. Methods

2.1. The data

The data for these experiments were obtained from a first-year primary school reader published circa 1950's (Hume). To keep this initial task simple, sentences with embedded structures (relative clauses) and a length of more than eight words were eliminated. The resulting corpus consists of 106 sentences ranging from three to eight words in length, average length 5.06 words. The words were converted to 10 lexical categories, including a sentence boundary marker. The categories, their abbreviations as used in the text and their percent frequencies are shown in Table 1. The resulting data consist of a string of 643 categories in 106 sentences. There are 62 distinct sentence sequences of which 43 occur only once, the rest being replicated. The maximum replication of any sequence is eight-fold. Where sequences, such as PR,VB,AR, are referred to in the text, AR is the current input, VB the previous input (at time step t-1) and PR the input at time step t-2.

2.2. The network

Elman simple recurrent networks (SRN), with ten input and ten output units representing the sparse coded lexical categories, were trained on the category sequence. The task was to predict the next lexical category given the current category.

Ingo Schellhammer, Joachim Diederich, Michael Towsey and Claudia Brugman (1998) **Knowledge Extraction and Recurrent Neural Networks: An Analysis of an Elman Network trained on a Natural Language Learning Task.** In D.M.W. Powers (ed.) *NeMLaP3/CoNLL98: New Methods in Language Processing and Computational Natural Language Learning*, ACL, pp 73-78.

The networks were trained by standard backpropagation with momentum and state unit activations were NOT reset to zero on presentation of a sentence boundary. Two networks were trained, one having two hidden units and the other nine, until prediction error stopped declining. The network with two hidden units had learned 51% of the training data and that with nine hidden units had learned 69% of the data. By way of comparison, 48%, 62%, 72% and 76% correct predictions could be obtained using bi-, tri-, 4- and 5-gram models of the training data respectively. At the end of training, the networks performed one pass through the data without learning in order to recover their hidden unit activations. Cluster analysis of the 642 output vectors was performed by graphical means for the two-hidden unit case and by *k*-means clustering for the nine-hidden unit case. Clusters from the latter case were used to prepare FSA's.

TABLE 1: Percent frequencies of the ten lexical categories in the text.

Lexical Category		*% frequency*
Article	AR	8%
Conjunction	CC	1%
Preposition	IN	7%
Adjective	JJ	4%
Noun	NN	30%
Pronoun	PR	10%
Possessive ('s)	PS	2%
Adverb	RB	1%
Verb	VB	20%
Sentence boundary	/S	17%

2.3. Cluster Analysis and Preparation of Finite State Automata

(i) *K-means* cluster analysis software was used to label the 642 hidden unit activation vectors with cluster numbers between 1 and k. Each vector was thus assigned to a state, S_i^t, where $1 \leq i \leq k$ and t uniquely identifies the time step for each member of cluster *i*.

(ii) For every current input, $\mathbf{x}^t$ and previous state, S_i^{t-1}, there is a transition to a new state, S_i^t with a resulting output, $\mathbf{o}^t$. A transition table was created from this data.

(iii) If the same input lead to more than one transition from a given state, the transition having highest frequency was chosen. Similarly, if any transition brought about by a given input, generated more than one possible output, the most frequent output was chosen.

(iv) The transition rules so derived were used to construct deterministic FSA's having *k* states corresponding to the *k* clusters. We generated ten FSA's with *k* taking values in the range 6 to 22.

(v) Each automaton was tested on the string of 643 categories used to train the original network. They were scored for total correct predictions, the fraction of missing transitions and score on the non-missing transitions.

(vi) In some experiments, low frequency transitions (having less than 5 occurrences) were pruned from the automaton and the resulting automaton again tested for its performance on the original data sequence. Missing transitions were handled by jumping to a predefined 'rescue' state and producing a predefined 'rescue' output. In the default instance, the rescue state was the state, whose preceding inputs had earliest position in the sentence. The rescue output was always NN, the category having highest frequency.

2.4. Weight Initialisation with Domain Knowledge

From an examination of bigram probabilities derived from the data sequence, it was determined that output categories NN and /S have the highest predictive rate. This knowledge can be used to initialise an Elman network with non-random weights in the expectation that training error should decline more rapidly than if the all the weights are initialised randomly. We initialised an Elman net having 11 hidden units with random weights between –0.1 and 0.1, and then manually set to a value of +4.0 some of the weights linking the hidden layer to the input units coding for NN and /S. We refer to these as the *set links.* In different trials, we set 0, 1, 5, 8, or 11 of both the NN and /S links in such a way as to minimise the number of hidden units having two set links. Zero set links means that none of the original random weights were changed.

3. Results

3.1. Graphical cluster analysis of the network having two hidden units.

Graphical cluster analysis for the 2-hidden unit case is shown in Figure 1. Clusters are labeled with the *current input.* There is marked separation of clusters representing the high frequency inputs, NN, VB, /S, PR and IN. There is overlap of those clusters representing low frequency inputs. Although only 51% of the training set was learned by the network, there is evidence of further clustering based on the current and previous inputs. For example, Figure 2 shows cluster formation when NN is the *current* input and either AR, NN, PR, PS, VB or /S is the *previous* input. The PR,NN sub-cluster could be further broken down into sub-sub-clusters, representing the three input sequences /S,PR,NN and IN,PR,NN and VB,PR,NN.

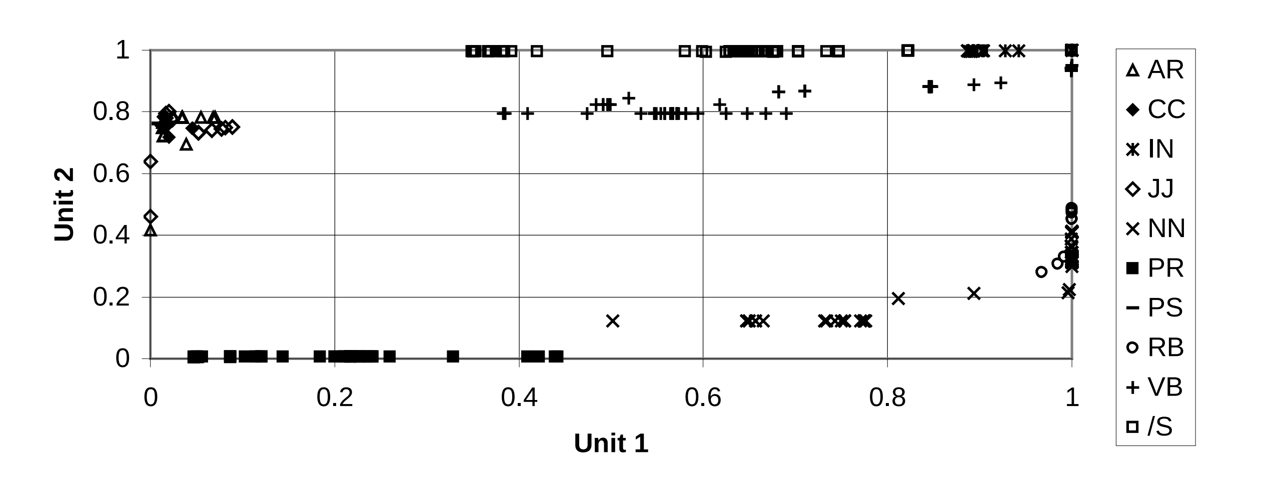

Figure 1: Hidden unit activations (of an Elman network with two hidden units) labeled according to which of the ten input categories is the current input producing that activation. The activations tend to be clustered according to the input. Clusters representing high frequency categories such as NN, VB and /S are more dispersed and broken into sub-clusters that represent both the current and previous inputs.

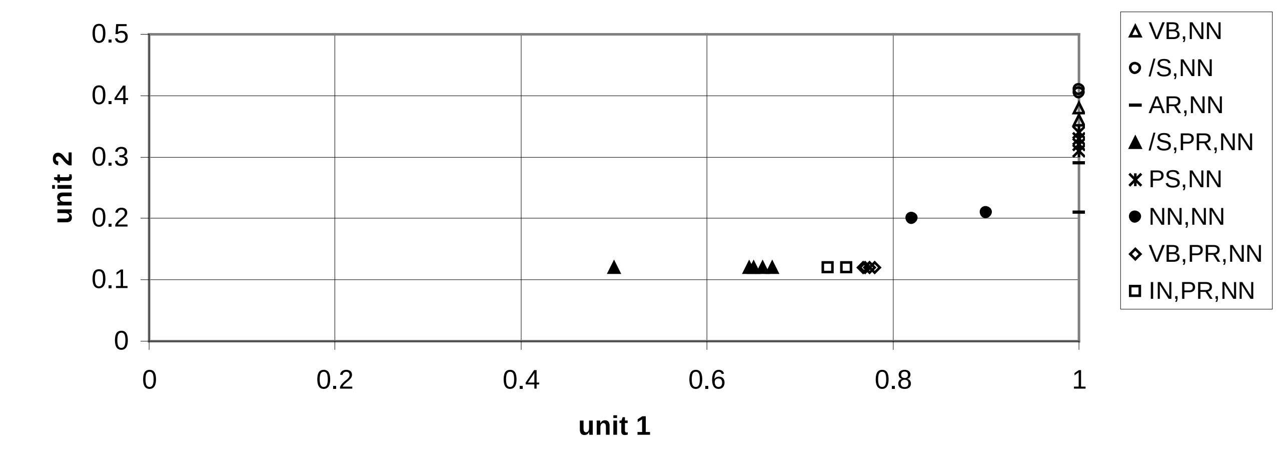

Figure 2: Hidden unit activations of an Elman network (with two hidden units only) when the current input category is NN and the previous category is either VB, /S, AR, PR, PS or NN.

3.2. Analysis of the FSA's

The performance of FSA's having 6 to 22 states is displayed in Table 2. The second column gives the total number of transitions permitted by the FSA. The third column gives the percent prediction score on the training data. Best score is 60% which compares with 69% of the training data learned by the original Elman network from which the hidden unit activations were obtained. The total prediction score tends to increase with the number of states.

The fourth column of Table 2 gives the percentage of the 642 transitions in the data not permitted by the FSA's. The number of missing transitions is small, in all but two cases less than 2%. When a *missing* transition occurs, the FSA defaults to a 'rescue' state. The percent correct predictions for non-missing transitions are shown in the rightmost column of Table 2. They are little different from the total scores in most cases, simply because the number of missing transitions is so few.

The transition diagram for the 8-state FSA is shown in Figure 3. The table in the top right of the figure shows: (i) the number of visits to each state when the FSA is tested, (ii) the percentage of correct predictions associated with a transition to that state and (iii) the average word-position in the sentence of the inputs leading to that state.

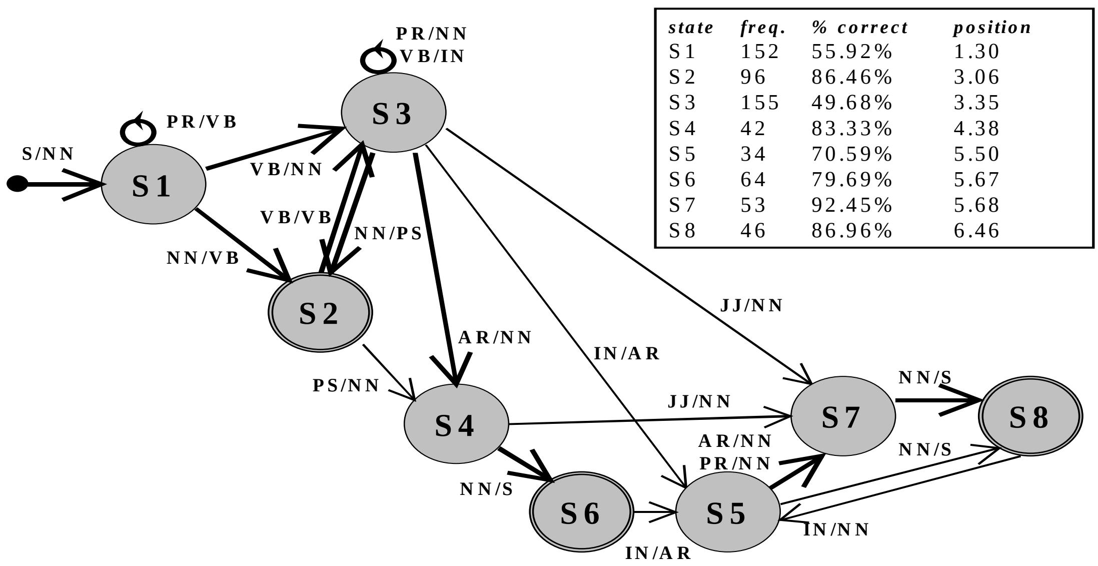

Figure 3: The FSA having 8 states. The double circle indicates an accept state. If the FSA is in an accept state and the input is /S (end of sentence) then it returns to the start state, S1, with the output of NN.

Transitions with thick arrows have a frequency count >20, transitions displayed with thin arrows have a frequency count of 5 to 20 and transitions with a frequency count <5 are not shown to preserve clarity. The states have been numbered in sequence according to the average word position of their associated inputs. For example states 2, 6 and 8 all occur following input of the NN category but they are distinguished in cluster analysis by the NN having an average word-position in the sentence of 3.1, 5.7 and 6.5 respectively.

Table 2
Performance of FSA's prepared from *k*-means cluster analysis of hidden unit activations.

No. of states	# of transitions	% total score	% missing transitions	% score on non-missing transitions
6	34	43	0.9	43
8	39	54	1.4	54
10	45	53	1.7	53
11	46	56	0.5	56
12	53	57	0.5	58
14	54	56	3.9	57
16	59	59	1.6	59
18	62	60	1.6	60
20	67	60	1.6	60
22	68	59	4.2	60

Table 3
The effect of removing low frequency transitions from an FSA having 10 states

No. of transitions	% total score	% missing transitions	% score on non-missing transitions
45	53	1.7	53
23	52	10.3	51

The states having highest correct prediction rate, S7 and S8, are associated with the ends of sentences. S7 is reached when the last category in a sentence is predicted to be NN and S8 occurs when the end-of-sentence is predicted.

Many of the transitions in the FSA's occur with low frequency and could be pruned with minimal loss of performance. For example, the FSA with ten states has 45 permitted transitions. When transitions having a frequency ≤5 are pruned, the number of missing transitions jumps from 1.7% to 10.28% but the prediction score drops only slightly from 53% to 51% (Table 3).

Finally we look at the effect of the state chosen as the rescue state for the FSA having 10 states. The default state is the state closest to the beginning of the sentence, in this case state 2.

The percent score of correct predictions is greater only in two other cases, that is when states 7 and 9 are used as the 'rescue' state. Changing the 'rescue' state also changes the number of transitions that the FSA does not recognise. However only in the case of rescue state 8 is this number less than for rescue state 2. It is apparent that a decrease in the number of missing transitions does not necessarily lead to a higher score.

3.3. Weight initialisation using domain knowledge

Setting links between the hidden layer and the NN and /S input units has a beneficial effect during the early stages of network training. As indicated by the faster initial decrease in prediction error, the optimum number of set links from inputs NN and /S was 5 or 8 (Figure 3).

Table 4
The effect of choice of 'rescue' state on the performance of the resulting 10-state FSA.

Rescue state	Average word position of inputs	% total score	% missing transitions	% score on non-missing transitions
S2	1.36	52	10.3	51
S5	2.94	50	15.3	53
S8	3.30	50	9.7	49
S3	4.18	49	12.8	48
S7	4.37	53	16.2	54
S10	5.53	52	17.3	55
S9	5.55	54	11.5	55
S4	5.92	49	19.8	53
S6	5.85	52	18.9	54
S1	6.48	49	19.8	53

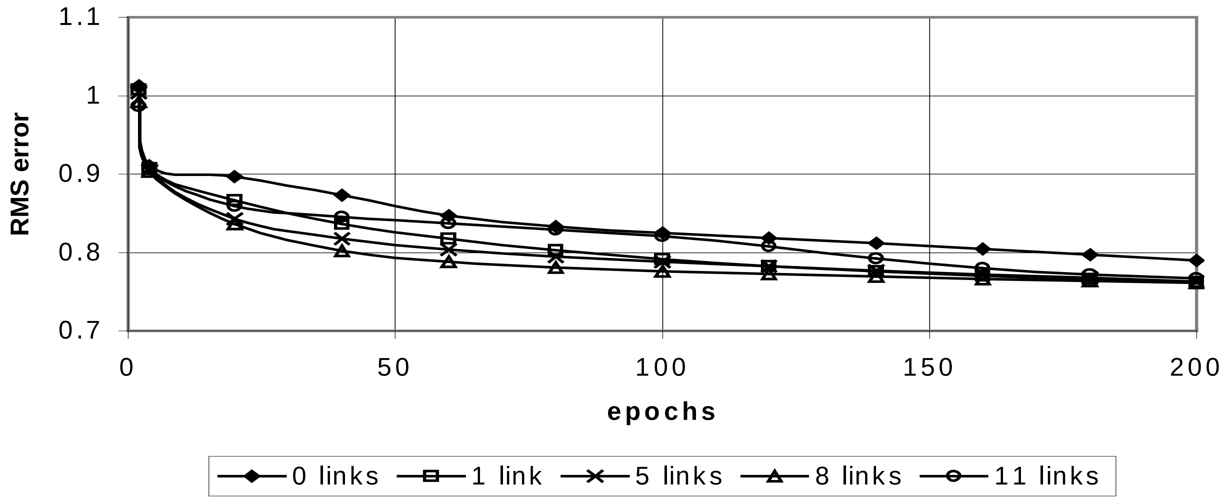

Figure 4: Output error of an Elman network over 200 training epochs following different weight initialisation procedures. '0-links' means that all links between hidden layer and input units are randomly initialised to values in [-0.1, 0.1]. The network has 11 hidden units. Therefore '11-links' means that every hidden unit has a *set link* (see methods section 2.4 for definition of this term) to the NN and the /S input units. 1, 5 and 8 links means that this number of hidden units has a *set link* to the NN and /S input units.

4. Discussion

Although an Elman network with two hidden units could learn only 51% of the training data, nevertheless graphical analysis reveals hierarchical clustering of hidden unit activations. There are clusters associated with each of the ten word categories (Figure 1), although clusters associated with low frequency inputs such as AR, CC and JJ tend to overlap. Clusters labeled with the high frequency inputs revealed obvious sub-clusters and sub-sub-clusters such as those shown in Figure 2. In other words, the network had acquired internal representations of temporal sequences of at least length 3. However because the hidden unit space had such low dimensionality, it could not be partitioned by the output layer to achieve accurate prediction.

An FSA with 18 states derived from *k*-means clustering of hidden unit activations scored 60% on the original training data (Table 2). This compares with a score of 69% by the original network and a score of 62% when predicting with a trigram model. Although in theory, the trigram model requires the calculation of

10,000 transition probabilities, it reduces to 42 transition rules. This compares with of 62 transitions rules incorporated into the 18-state FSA. Thus the trigram model is a more compact definition of the grammar. However, low frequency transitions can be trimmed from the FSA's with minimal loss of performance as is demonstrated for the 10-state FSA in Table 3.

Correct choice of the 'rescue' state is important for the efficient performance of an FSA because it determines the FSA's ability to pick up the sentence structure after a missing transition. In order to automate the production of FSA's following cluster analysis, we require a heuristic for the choice of 'rescue' state. Our initial choice, *that state whose inputs on average are closest to the beginning of the sentence,* seems to be a reasonable heuristic in the absence of other information (Table 4). Likewise, choosing the highest frequency category (in our case, NN) as the 'rescue' output is also confirmed by our results because the FSA scores achieved on non-missing transitions are not much better than the total scores, despite 10-20% of missing transitions (Table 4).

The use of domain knowledge, such as category frequencies, to bias weight initialisation is successful in reducing error faster in the early stages of learning. Of course if training is continued for long enough, then any memory of the initial bias will be erased. Best results were achieved when five links were set (such that no hidden unit had a set link to both the NN and /S inputs) or eight links were set (such that only five hidden units had set links to both the NN and /S inputs).

5. Conclusions

This study has demonstrated one method for extracting the knowledge encoded in a trained neural network. Quite often knowledge extracted from neural networks is in the form of propositional rules (Andrews et al., 1995) but these are not always the most appropriate format for explication of network learning. For example where the network has been required to induce a grammar, cluster analysis of hidden unit activations and preparation of an FSA is a powerful technique to explicate the learned grammar. However, for this particular task, there is a trade-off between comprehensibility of the FSA (fewer states means more comprehensible) and its predictive performance compared to the original neural network. In these experiments an FSA with 18-states performed almost as well as a trigram model. The trigram model had the advantage of compactness, but the FSA had the advantage of comprehensibility.

6. References

Andrews, R., Diederich, J., & Tickle A.B. (1995). A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, *8(6)*, 373-389

Elman, J.L. (1990). Finding Structure in Time. *Cognitive Science 14*, 179-211.

Giles, C.L. & Omlin, C.W. (1993a). Rule refinement with recurrent neural networks. *Proceedings. of the IEEE International Conference on Neural Networks* (pp. 801-806). San Francisco, CA.

Giles, C.L. & Omlin, C.W. (1993b). Extraction, insertion, and refinement of symbolic rules in dynamically driven recurrent networks. *Connection Science*, 5(3 & 4), 307-328.