Appears in Symbolic, Connectionist, and Statistical Approaches to Learning for Natural Language Processing, Lecture Notes in AI, edited by Stefan Wermter, Ellen Riloff and Gabriele Scheler, Springer Verlag, New York, pp. 33–47, 1996.

Natural Language Grammatical Inference: A Comparison of Recurrent Neural Networks and Machine Learning Methods

Steve Lawrence[1,2*], Sandiway Fong[1], C. Lee Giles[1†‡]
lawrence@research.nj.nec.com, {sandiway,giles}@research.nj.nec.com

[1] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540
[2] Electrical and Computer Engineering, University of Queensland, St. Lucia 4072, Australia

Abstract

We consider the task of training a neural network to classify natural language sentences as grammatical or ungrammatical, thereby exhibiting the same kind of discriminatory power provided by the Principles and Parameters linguistic framework, or Government and Binding theory. We investigate the following models: feed-forward neural networks, Frasconi-Gori-Soda and Back-Tsoi locally recurrent neural networks, Williams and Zipser and Elman recurrent neural networks, Euclidean and edit-distance nearest-neighbors, and decision trees. Non-neural network machine learning methods are included primarily for comparison. We find that the Elman and Williams & Zipser recurrent neural networks are able to find a representation for the grammar which we believe is more parsimonious. These models exhibit the best performance.

1 Motivation

1.1 Representational Power of Recurrent Neural Networks

Natural language has traditionally been handled using symbolic computation and recursive processes. The most successful stochastic language models have been based on finite-state descriptions such as *n*-grams or hidden Markov models. However, finite-state models cannot represent hierarchical structures as found in natural language[1] (Pereira & Schabes 1992). In the past few years several recurrent neural network (RNN) architectures have emerged which have been used for grammatical inference. RNNs have been used for several smaller natural language problems, e.g. papers using the Elman network for natural language tasks include: (Stolcke 1990, Allen 1983, Elman 1984, Harris & Elman 1984, St. John & McClelland 1990). Neural network models have been shown to be able to account for a variety of phenomena in phonology (Gasser & Lee 1990, Hare 1990, Touretzky 1989*a*, Touretzky 1989*b*), morphology (Hare, Corina & Cottrell 1989, MacWhinney, Leinbach, Taraban & McDonald 1989) and role assignment (Miikkulainen & Dyer 1989, St. John & McClelland 1990). Induction of simpler grammars has been addressed often – e.g. (Watrous & Kuhn 1992, Giles, Miller, Chen, Chen, Sun & Lee 1992) on learning Tomita languages. There has been some interest in learning more than regular grammars with RNNs. Jordan and others (Pollack 1990, Berg 1992, Sperduti, Starita & Goller 1995) have used recursive auto-associative distributed memories (RAAMs) in RNNs. Others (Das, Giles & Sun 1992, Sun, Giles, Chen & Lee 1993, Zeng, Goodman & Smyth 1994) have used RNNs tied to external trainable

[*] http://www.neci.nj.nec.com/homepages/lawrence
[†] Also with the Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.
[‡] http://www.neci.nj.nec.com/homepages/giles.html
[1] The inside-outside re-estimation algorithm is an extension of hidden Markov models intended to be useful for learning hierarchical systems. The algorithm is currently only practical for relatively small grammars (Pereira & Schabes 1992).

stacks. The grammars learned were not large. Our task differs from these in that the grammar is considerably more complex. Recently large regular grammars have been learned by RNNs (Giles, Horne & Lin 1995, Clouse, Giles, Horne & Cottrell 1994). However, these grammars have unusual properties when implemented as sequential machines in the sense that they have little logic.

It has been shown that RNNs have the representational power required for hierarchical solutions (Elman 1991), and that they are Turing equivalent (Siegelmann & Sontag 1995). The RNNs investigated in this paper constitute complex, dynamical systems. Pollack (Pollack 1991) points out that Crutchfield and Young (Crutchfield & Young 1989) have studied the computational complexity of dynamical systems reaching the onset of chaos via period-doubling. They have shown that these systems are not regular, but are finitely described by indexed context-free-grammars. Several modern computational linguistic grammatical theories fall in this class (Joshi 1985, Pollard 1984).

1.2 Language and Its Acquisition

Certainly one of the most important questions for the study of human language is: How do people unfailingly manage to acquire such a complex rule system? A system so complex that it has resisted the efforts of linguists to date to adequately describe in a formal system (Chomsky 1986)? Here, we will provide a couple of examples of the kind of knowledge native speakers often take for granted.

For instance, any native speaker of English knows that the adjective *eager* obligatorily takes a complementizer *for* with a sentential complement that contains an overt subject, but that the verb *believe* cannot. Moreover, *eager* may take a sentential complement with a non-overt, i.e. an implied or understood, subject, but *believe* cannot:[2]

*I am eager John to be here	I believe John to be here
I am eager for John to be here	*I believe for John to be here
I am eager to be here	*I believe to be here

Such grammaticality judgments are sometimes subtle but unarguably form part of the native speaker's language competence. In other cases, judgment falls not on acceptability but on other aspects of language competence such as interpretation. Consider the reference of the embedded subject of the predicate *to talk to* in the following examples:

John is too stubborn for Mary to talk to
John is too stubborn to talk to
John is too stubborn to talk to Bill

In the first sentence, it is clear that *Mary* is the subject of the embedded predicate. As every native speaker knows, there is a strong contrast in the co-reference options for the understood subject in the second and third sentences despite their surface similarity. In the third sentence, *John* must be the implied subject of the predicate *to talk to*. By contrast, *John* is understood as the object of the predicate in the second sentence, the subject here having arbitrary reference; in other words, the sentence can be read as *John is too stubborn for some arbitrary person to talk to John*. The point we would like to emphasize here is that the language faculty has impressive discriminatory power, in the sense that a single word, as seen in the examples above, can result in sharp differences in acceptability or alter the interpretation of a sentence considerably. Furthermore, the judgments shown above are robust in the sense that virtually all native speakers will agree with the data.

In the light of such examples and the fact that such contrasts crop up not just in English but in other languages (for example, the *stubborn* contrast also holds in Dutch), some linguists (chiefly Chomsky (Chomsky 1981)) have hypothesized that it is only reasonable that such knowledge is only partially acquired: the lack of variation found across speakers, and indeed, languages for certain classes of data suggests that there exists a fixed component of the

[2] As is conventional, we use the asterisk to indicate ungrammaticality in these examples.

Table 1: Parts-of-speech

Category	Examples
Nouns (N)	*John*, *book* and *destruction*
Verbs (V)	*hit*, *be* and *sleep*
Adjectives (A)	*eager*, *old* and *happy*
Prepositions (P)	*without* and *from*
Complementizer (C)	*that* or *for* as in *I thought* that . . . or *I am eager* for . . .
Determiner (D)	*the* or *each* as in the *man* or each *man*
Adverb (Adv)	*sincerely* or *why* as in *I* sincerely *believe* . . . or Why *did John want* . . .
Marker (Mrkr)	possessive *'s*, *of*, or *to* as in *John*'s *mother*, *the destruction* of . . . , or *I want* to *help* . . .

language system. In other words, there is an innate component of the language faculty of the human mind that governs language processing. All languages obey these so-called universal principles. Since languages do differ with regard to things like subject-object-verb order, these principles are subject to parameters encoding systematic variations found in particular languages. Under the innateness hypothesis, only the language parameters plus the language-specific lexicon are acquired by the speaker; in particular, the principles are not learned. Based on these assumptions, the study of these language-independent principles has become known as the Principles-and-Parameters framework, or Government-and-Binding (GB) theory.

In this paper, we ask the question: Can a neural network be made to exhibit the same kind of discriminatory power on the data GB-linguists have examined? More precisely, the goal of the experiment is to train a neural network from scratch, i.e. without the bifurcation into learned vs. innate components assumed by Chomsky, to produce the same judgments as native speakers on the sharply grammatical/ungrammatical pairs of the sort discussed above.

The remainder of the paper is organized as follows. In Sect. 2 we introduce our dataset and describe our problem. In sections 3 to 6 we describe the models used and our experimental methodology. We present our results in Sect. 7 and discuss techniques we used to improve the results in Sect. 8. Full details on a successful simulation are given in Sect. 9 and we draw conclusions in Sect. 10.

2 Data

Our primary data consists of 552 English positive and negative examples taken from an introductory GB-linguistics textbook by Lasnik and Uriagereka (Lasnik & Uriagereka 1988). Most of these examples are organized into minimal pairs like the example *I am eager for John to be here/*I am eager John to be here* that we have seen earlier. We note here that the minimal nature of the changes involved suggests that our dataset may represent an especially difficult task. Due to the small sample size, the raw data was first converted (using an existing parser) into the major syntactic categories assumed under GB-theory. Table 1 summarizes the parts-of-speech that were used.

The part-of-speech tagging represents the sole grammatical information supplied to the neural network about particular sentences in addition to the grammaticality status. A small but important refinement that was implemented was to include subcategorization information for the major predicates, namely nouns, verbs, adjectives and prepositions. (Our experiments showed that adding subcategorization to the bare category information improved the performance of the neural networks.) For example, an intransitive verb such as *sleep* would be placed into a different class from the obligatorily transitive verb *hit*. Similarly, verbs that take sentential complements or double objects such as *seem*, *give* or *persuade* would be representative of other classes.[3] Flushing out the subcategorization requirements along these lines for lexical items in the training set resulted in 9 classes for verbs, 4 for nouns and adjectives, and 2 for prepositions. Examples of the input data are shown in Table 2. We note here that tagging was done in a completely context-free manner. Obviously, a word, e.g. *to*, may be part of more than one part-of-speech. The tagger being part of a larger parsing system is capable of assigning the correct parts-of-speech, but no disambiguation was done to provide a greater challenge.

Table 2: Examples of part-of-speech tagging

Sentence	Encoding	Grammatical Status
I am eager for John to be here	n4 v2 a2 c n4 v2 adv	1
	n4 v2 a2 c n4 p1 v2 adv	1
I am eager John to be here	n4 v2 a2 n4 v2 adv	0
	n4 v2 a2 n4 p1 v2 adv	0

3 Experimental Methodology

We divide the data up into a training set and a test set. A window of a fixed number of symbols is used as input to the models and the width of this window is varied for different simulations. The window is moved from the start to the end of each sentence in temporal order, and a classification is obtained at the final step. For those models which cannot form internal states, it does not make sense to use an input window which is smaller than the length of the sentence, although we do present some results in this case for comparison purposes.

For input to the neural networks and nearest-neighbor models, the 23 part-of-speech symbols were encoded into a fixed length window made up of segments containing eight separate inputs, corresponding to the classifications noun, verb, adjective, etc. Sub-categories of the classes were linearly encoded into each input in a manner demonstrated by the specific values for the noun input: Not a noun = 0, noun class 1 = 0.5, noun class 2 = 0.667, noun class 3 = 0.833, noun class 4 = 1. The linear order was defined using our judgment of the similarity between the various sub-categories. Two outputs were used in the neural networks, corresponding to grammatical and ungrammatical classifications. A confidence criteria was used: $y_{max} \times (y_{max} - y_{min})$[4]. There is some contradiction in the data due to the ambiguity of the part-of-speech conversion. We decided to remove the contradictory data for the results presented here.

It is important to ask whether results are statistically significant. Our results are based on multiple training/test set partitions and multiple random seeds (for those models which use them). We equalized the number of positive and negative examples in the training and test sets in order to ensure that the models cannot produce results based on the ratio of positive to negative examples. We have also used a set of Japanese control data. Japanese is at the opposite end of the language spectrum when compared to English and we expect a model trained on the English data to perform poorly on the Japanese data. Indeed, all models attain 50% or less correct classification on average for the Japanese data.

More details can be found in Sect. 9 and (Lawrence, Giles & Fong 1995).

[3] Following classical GB theory, these classes are synthesized from the theta-grids of individual predicates via the Canonical Structural Realization (CSR) mechanism of Pesetsky (Pesetsky 1982).

[4] For an output range of 0 to 1 and softmax outputs.

4 Nearest-Neighbors

In the nearest-neighbors technique, the nearest-neighbors to a test sentence are found using a similarity measure. The class of the test sentence is inferred from the classes of the neighbors. We investigated the following similarity measures:

1. *Euclidean distance.* The neighbors are found based on their Euclidean distance from the test sentence in the space created by the input encoding ($\sqrt{\sum_{i=1}^{n}(y_i - d_i)^2}$). As expected, models with a small temporal window did not achieve significantly greater than 50% correct classification. However, models with a large temporal window (near the size of the longest sentences) achieved up to 65% correct classification on average.

2. *Edit distance.* In edit-distance computation a cost is assigned for inserting, deleting, and substituting symbols in a sequence. Dynamic programming can be used to calculate the cost of transforming one sequence into another [5]. We were unable to attain greater than 55% correct classification on average for the test set. Although we expect careful selection of the cost table to improve performance, analysis of the operation of the method leads us to believe that it will never obtain very good performance[6].

5 Decision Trees

For comparison purposes, we tested the C4.5 decision tree induction algorithm by Ross Quinlan (Quinlan 1993). Decision tree methods construct a tree which partitions the data at each level in the tree based on a particular feature of the data. C4.5 only deals with strings of constant length, hence we used an input window corresponding to the longest string. We expect that significantly more data would be required to match the performance of the RNNs due to the position dependence created by the fixed input window. We used the standard settings in the C4.5 package for pruning, etc. We obtained 60% correct classification performance on the test data on average.

6 Neural Networks

Our primary interest was to train an RNN to form internal states in order to create an automata which may describe the grammar in a more parsimonious manner, and hence generalize better to unseen examples. However, we tested feedforward networks with time delayed inputs for comparison as well. We tested the following models: Multi-layer perceptron (MLP), Frasconi-Gori-Soda (FGS), Back-Tsoi FIR, Williams and Zipser, and Elman neural networks. A brief description of each model follows

1. *Multi-layer perceptron.* The output of a neuron is computed using[7]

[5]Sequences of length zero up to the actual sequence length are considered. The following equations are used iteratively to calculate the distances ending in the distance between the two complete sequences. i and j range from 0 to the length of the respective sequences and the superscripts denote sequences of the corresponding length. For more details see (Kruskal 1983).

$$d(\mathbf{a}^i, \mathbf{b}^j) = \min \begin{cases} d(\mathbf{a}^{i-1}, \mathbf{b}^j + w(a_i, 0) & \text{deletion of} a_i \\ d(\mathbf{a}^{i-1}, \mathbf{b}^{j-1}) + w(a_i, b_j) & b_j \text{ replaces} a_i \\ d(\mathbf{a}^i, \mathbf{b}^{j-1}) + w(0, b_j) & \text{insertion of} b_j \end{cases}$$

[6]Consider how to define the cost for deleting a noun without knowing the context in which it appears.

[7]where y_k^l is the output of neuron k in layer l, N_l is the number of neurons in layer l, w_{ki}^l is the weight connecting neuron k in layer l to neuron i in layer $l-1$, $y_0^l = 1$ (bias), and f is commonly a sigmoid function.

$$y_k^l = f\left(\sum_{i=0}^{N_{l-1}} w_{ki}^l y_i^{l-1}\right) \tag{1}$$

2. *Frasconi-Gori-Soda locally recurrent networks.* A network with local feedback connection around the hidden nodes as described in (Frasconi, Gori & Soda 1992).
3. *Back-Tsoi FIR.* A multi-layer perceptron network with an FIR filter and a gain term included in every synapse as described in (Back & Tsoi 1991).
4. *Williams and Zipser.* A fully connected recurrent network where all nodes are connected to all other nodes as described in (Williams & Zipser 1989).
5. *Elman (Elman 1990, Elman 1991).* A recurrent network where each hidden layer node has a feedback connection to all other hidden layer nodes. The feedback connections are trainable and we use full backpropagation through time for training instead of the truncated version used by Elman.

The size of the neural networks (the number of hidden nodes) was determined based on the goal that the networks should be large enough to learn the training data in a reasonable time but no larger (larger networks lead to easy creation of a lookup table by the model and poor generalization – this is known as overfitting).

7 Results

Initially, partial success was only obtained with models employing a large temporal input window. We were unable to train the networks using a small temporal window although it is theoretically possible. We experienced difficulty because the gradient descent search technique used became stuck in local minima. We are interested in training RNNs with small temporal input windows because these are the only networks which are forced to form internal states in order to learn successfully – they are forced to look for solutions which may be more parsimonious and generalize better. After adding techniques designed to avoid local minima we were able to train an RNN with sequences of the last two words as input to give 100% correct classification on the training data. This performance was obtained with an Elman RNN. Generalization on the test data resulted in 74% correct classification on average. This is better than the performance obtained using any of the other methods, however it is still quite low. The available data is quite sparse and we expect increased generalization performance as the amount of data increases (as well as increased difficulty in training). Additionally, the dataset has been hand-designed by GB linguists to cover a range of grammatical structures and it is likely that the separation into the training and test sets creates a test set containing many grammatical structures that are not covered in the training set.

Tables 3 and 4 summarize the results obtained with the various methods. The locally recurrent networks (Frasconi-Gori-Soda, Back-Tsoi) did not perform significantly better than the standard multi-layer perceptron. Using a large temporal input window, the neural network models were able to attain 100% correct classification performance on the training data and 65% correct classification on the test data. The nearest-neighbor and decision tree methods did not exceed this performance level. Using a small temporal input window, no model except the Elman RNN exceeded 55% correct classification on the test data. The Elman network was able to attain 74% correct classification on the test data and the W&Z network was able to attain 71%. In order to make the number of weights in each architecture approximately equal we have used only single word inputs for the W&Z model but two word inputs for the others. This reduction in dimensionality for the W&Z network improved performance.

Table 3: Percentage correct classification for the training data.

TRAIN	large window	small window
MLP	100	55
FGS	100	56
BT-FIR	100	56
Elman	100	**100**
W&Z	100	**92**

Table 4: Percentage correct classification for the test data.

TEST	large window	small window
Edit-distance	55	N/A
Euclidean	65	55
Decision trees	60	N/A
MLP	63	54
FGS	65	59
BT-FIR	64	54
Elman	65	**74**
W&Z	59	**71**

8 Gradient Descent

We have used backpropagation-through-time [8] (Williams & Zipser 1990) to train the globally-recurrent networks [9] and the gradient descent algorithm described by the authors for the FGS and Back-Tsoi FIR networks. The error surface of a multilayer network is generally non-convex, non-quadratic, and often has large dimensionality. We found the standard gradient descent algorithms to be impractical for our problem. We employed the techniques described below for improving convergence. Although these techniques are heuristic and reduce the elegance of the solution, they are motivated by analyzing the operation of the algorithm and are applicable to many other problems.

1. Detection of Significant Error Increases. If the NMSE increases significantly during training then the network weights are restored from a previous epoch and are perturbed to prevent updating to the same point. We have found this technique to increase robustness of the algorithm when using learning rates large enough to help escape local minima, particularly in the case of the Williams & Zipser network.

2. Target outputs. Target outputs were 0.1 and 0.9 using the logistic activation function and -0.8 and 0.8 using the $tanh$ activation function. This helps avoid saturating the sigmoid function. If targets were set to the asymptotes of the sigmoid this would tend to: a) drive the weights to infinity, b) cause outlier data to produce very large gradients due to the large weights, and c) produce binary outputs even when incorrect – leading to decreased reliability of the confidence measure.

3. Stochastic updating. In stochastic update parameters are updated after each pattern presentation, whereas in true gradient descent (often called "batch" updating) gradients are accumulated over the complete training set. Batch

[8] Backpropagation-through-time extends backpropagation to include temporal aspects and arbitrary connection topologies by considering an equivalent feedforward network created by unfolding the recurrent network in time.

[9] Real-time recurrent learning (Williams & Zipser 1989) was also tested but did not show any significant convergence for our problem.

update attempts to follow the true gradient, whereas stochastic is similar to adding noise to the true gradient – this noise can help the algorithm avoid local minima. We found that stochastic update produced significantly better results.

4. Learning rate schedule. Relatively high learning rates are typically used in order to help avoid slow convergence and local minima. However, a constant learning rate results in significant parameter and performance fluctuation during the entire training cycle such that the performance of the network can alter significantly from the beginning to the end of the final epoch. Moody and Darkin have proposed "search then converge" learning rate schedules of the form (Darken & Moody 1991): $\eta(t) = \eta_0 / (1 + t/\tau)$ where $\eta(t)$ is the learning rate at time t, η_0 is the initial learning rate, and τ is a constant. We have found that the learning rate during the final epoch still results in considerable parameter fluctuation[10] and hence we have added an additional term (see section 9) to further reduce the learning rate over the final epochs. We have found the use of learning rate schedules to improve performance considerably.

In addition to these techniques, we also tried the following, but were unable to obtain improved performance.

1. Sigmoid functions. Symmetric sigmoid functions (e.g. tanh) often improve convergence over the standard logistic function. For our particular problem we found the difference was minor.

2. Sectioning of the training data. We investigated dividing the training data into subsets. Initially, only one of these subsets was used for training. After 100% correct classification was obtained or a pre-specified time limit expired, an additional subset was added to the "working" set. This continued until the working set contained the entire training set. These trials were performed with the data ordered alphabetically, which enabled the networks to focus on the simpler data first. Elman suggests that the initial training constrains later training in a useful way (Elman 1991). The use of sectioning has consistently decreased performance in our case.

3. Cost function. The relative entropy cost function has received particular attention (Haykin 1994) and has a natural interpretation in terms of learning probabilities (Kullback 1959). We investigated using both quadratic ($E = \frac{1}{2}\sum_k (y_k - d_k)^2$) and relative entropy cost functions ($E = \sum_k [\frac{1}{2}(1 + y_k)\ log \frac{1+y_k}{1+d_k} + \frac{1}{2}\ (1 - y_k)\ log \frac{1-y_k}{1-d_k}]$): where y and d correspond to the actual and desired output values, and k ranges over the outputs (and also the patterns for batch update). We found the quadratic cost function to provide better performance. A possible reason for this is that the use of the entropy cost function leads to an increased variance of weight updates and therefore decreased robustness in parameter updating.

9 Simulation Details

Full details for the best performing model, the Elman network, are given below. For the remaining models, variations in specific implementation details (e.g. the number of hidden nodes) did not make a significant difference. The results reported in this paper are for an initial learning rate of 0.2 and 20 hidden nodes.

The network contained three layers including the input layer. The hidden layer contained 20 nodes. Each hidden layer node had a recurrent connection to all other hidden layer nodes. The network was trained for a total of 1 million stochastic updates. All inputs and outputs were within the range zero to one. Bias inputs were used. The best of 50 random weight sets was chosen based on training set performance. Weights were initialized as shown in Haykin (Haykin 1994). Targets outputs were 0.1 and 0.9 using the logistic output activation function. The quadratic cost function was used. The search then converge learning rate schedule used was $\eta = \frac{\eta_0}{\frac{n}{N/2} + \frac{c_1}{max\left(1, (c_1 - \frac{max(0, c_1(n - c_2 N))}{(1-c_2)N}\right)}}$ where η = learning rate, η_0 = initial learning rate = 0.2, N = total training epochs, n = current training epoch, $c_1 = 50$, $c_2 = 0.65$. The training set consisted of 373 non-contradictory examples. The English test set consisted of 100 non-contradictory samples and the Japanese test set consisted of 119 non-contradictory samples. 74% correct classification was obtained on the English test set and 53% correct classification was obtained on the Japanese test set. The total training time was around 4 hours on a Sun Sparc 10 workstation.

[10]NMSE results which are obtained over an epoch involving stochastic update can be misleading. We have been surprised to find quite significant difference in these on-line NMSE calculations compared to a static calculation even if the algorithm appears to have converged.

10 Conclusions

We investigated the use of feed-forward neural networks, Frasconi-Gori-Soda and Back-Tsoi locally recurrent neural networks, Williams and Zipser and Elman recurrent neural networks, Euclidean and edit-distance nearest-neighbors, and decision trees for classifying natural language sentences as grammatical or ungrammatical, thereby exhibiting the same kind of discriminatory power provided by the Principles and Parameters linguistic framework, or Government-and-Binding theory. From best to worst performance, the architectures are roughly: Elman, W&Z, FGS, Euclidean distance nearest neighbors, MLP, C4.5 decision trees, and edit-distance nearest neighbors. Theoretically, a W&Z network with the same number of nodes should have at least the same representational power as an Elman network, yet the Elman network provides better performance. Investigation shows that this is due to the more complex error surface of the W&Z architecture.

Are the methods really learning anything? Nearest-neighbors, decision trees, feedforward networks do not learn parsimonious representations of the grammar – they work by finding statistically close matches in the training data. They are expected to require a much larger amount of data for similar performance. On the other hand, recurrent neural networks do learn a grammar. 100% correct classification of the training data is not possible using only a small temporal input window without forming internal states.

We have shown that both Elman and W&Z recurrent neural networks are able to learn an appropriate grammar for discriminating between the sharply grammatical/ungrammatical pairs used by GB-linguists. The recurrent neural networks considered here posses the required representational power for the task considered. However, difficulty lies in finding the optimum parameters (weights). The standard training algorithms were unable to find a solution to our problem. We were able to find a solution by adding techniques aimed at improving the convergence of the training algorithm.

Current generalization performance is not very high, however, the results appear significant considering the nature of the dataset (sparse and grammatical structures differ between the training and test sets). We plan to generate more data – we expect better generalization performance, however we also expect training to be more difficult. Hence, we need to continue to address the convergence of the training algorithms. We believe that the basic assumptions of smoothness in the required function approximation and the nature of parameter updating provide opportunities for improvement (as demonstrated by the techniques described within). The grammar learnt by the recurrent networks could be extracted and examined (Giles et al. 1992, Watrous & Kuhn 1992). Further progress can be made by continuing to address the convergence of the backpropagation algorithm. However, we envisage a point after which advances will depend on considering the connectionist treatment of hierarchical representations.

References

Allen, R. B. (1983), Sequential connectionist networks for answering simple questions about a microworld, *in* '5th Annual Proceedings of the Cognitive Science Society', pp. 489–495.

Back, A. & Tsoi, A. (1991), 'FIR and IIR synapses, a new neural network architecture for time series modeling', *Neural Computation* **3**(3), 375–385.

Berg, G. (1992), A connectionist parser with recursive sentence structure and lexical disambiguation, *in* 'Proceedings AAAI', pp. 32–37.

Chomsky, N. (1981), *Lectures on Government and Binding*, Foris Publications.

Chomsky, N. (1986), *Knowledge of Language: Its Nature, Origin, and Use*, Prager.

Clouse, D., Giles, C. L., Horne, B. & Cottrell, G. (1994), Learning large DeBruijn automata with feed-forward neural networks, Technical Report CS94-398, Computer Science and Engineering, University of California at San Diego, La Jolla, CA.

Crutchfield, J. P. & Young, K. (1989), Computation at the onset of chaos, *in* W. Zurek, ed., 'Complexity, Entropy and the Physics of Information', Addison-Wesley, Reading, MA.

Darken, C. & Moody, J. (1991), Note on learning rate schedules for stochastic optimization, *in* R. Lippmann, J. Moody & D. S. Touretzky, eds, 'Advances in Neural Information Processing Systems', Vol. 3, Morgan Kaufmann, San Mateo, CA, pp. 832–838.

Das, S., Giles, C. L. & Sun, G. (1992), Learning context-free grammars: Limitations of a recurrent neural network with an external stack memory, *in* 'Proceedings of The Fourteenth Annual Conference of the Cognitive Science Society', Morgan Kaufmann Publishers, San Mateo, CA, pp. 791–795.

Elman, J. (1984), Structured representations and connectionist models, *in* '6th Annual Proceedings of the Cognitive Science Society', pp. 17–25.

Elman, J. (1990), 'Finding structure in time', *Cognitive Science* **14**, 179–211.

Elman, J. (1991), 'Distributed representations, simple recurrent networks, and grammatical structure', *Machine Learning* **7**(2/3), 195–226.

Frasconi, P., Gori, M. & Soda, G. (1992), 'Local feedback multilayered networks', *Neural Computation* **4**(1), 120–130.

Gasser, M. & Lee, C. (1990), Networks that learn phonology, Technical report, Computer Science Department, Indiana University.

Giles, C. L., Horne, B. & Lin, T. (1995), 'Learning a class of large finite state machines with a recurrent neural network', *Neural Networks* . In press.

Giles, C. L., Miller, C., Chen, D., Chen, H., Sun, G. & Lee, Y. (1992), 'Learning and extracting finite state automata with second-order recurrent neural networks', *Neural Computation* **4**(3), 393–405.

Hare, M. (1990), The role of similarity in Hungarian vowel harmony: A connectionist account, Technical Report CRL 9004, Centre for Research in Language, University of California, San Diego.

Hare, M., Corina, D. & Cottrell, G. (1989), Connectionist perspective on prosodic structure, Technical Report CRL Newsletter Volume 3 Number 2, Centre for Research in Language, University of California, San Diego.

Harris, C. L. & Elman, J. (1984), Representing variable information with simple recurrent networks, *in* '6th Annual Proceedings of the Cognitive Science Society', pp. 635–642.

Haykin, S. (1994), *Neural Networks, A Comprehensive Foundation*, Macmillan, New York, NY.

Joshi, A. K. (1985), Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions?, *in* L. K. D. R. Dowty & A. M. Zwicky, eds, 'Natural Language Parsing', Cambridge University Press, Cambridge.

Kruskal, J. B. (1983), An overview of sequence comparison, *in* D. Sankoff & J. B. Kruskal, eds, 'Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison', Addison-Wesley, Reading, Massachusetts.

Kullback, S. (1959), *Information Theory and Statistics*, Wiley, New York.

Lasnik, H. & Uriagereka, J. (1988), *A Course in GB Syntax: Lectures on Binding and Empty Categories*, MIT Press, Cambridge, MA.

Lawrence, S., Giles, C. L. & Fong, S. (1995), On the applicability of neural network and machine learning methodologies to natural language processing, Technical Report UMIACS-TR-95-64 and CS-TR-3479, Institute for Advanced Computer Studies, University of Maryland, College Park MD 20742, `http://www.neci.nj.nec.-com/homepages/lawrence/papers/nl-tr95/nl-tr95.html`.

MacWhinney, B., Leinbach, J., Taraban, R. & McDonald, J. (1989), 'Language learning: cues or rules?', *Journal of Memory and Language* **28**, 255–277.

Miikkulainen, R. & Dyer, M. (1989), Encoding input/output representations in connectionist cognitive systems, *in* D. S. Touretzky, G. E. Hinton & T. J. Sejnowski, eds, 'Proceedings of the 1988 Connectionist Models Summer School', Morgan Kaufmann, Los Altos, CA, pp. 188–195.

Pereira, F. & Schabes, Y. (1992), Inside-outside re-estimation from partially bracketed corpora, *in* 'Proceedings of the 30th annual meeting of the ACL', Newark, pp. 128–135.

Pesetsky, D. M. (1982), Paths and Categories, PhD thesis, MIT.

Pollack, J. (1990), 'Recursive distributed representations', *Artificial Intelligence* **46**, 77–105.

Pollack, J. (1991), 'The induction of dynamical recognizers', *Machine Learning* **7**, 227–252.

Pollard, C. (1984), Generalised context-free grammars, head grammars and natural language, PhD thesis, Department of Linguistics, Stanford University, Palo Alto, CA.

Quinlan, R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, California.

Siegelmann, H. & Sontag, E. (1995), 'On the computational power of neural nets', *Journal of Computer and System Sciences* **50**(1), 132–150.

Sperduti, A., Starita, A. & Goller, C. (1995), Learning distributed representations for the classification of terms, *in* 'Proceedings of the International Joint Conference on Artificial Intelligence', pp. 509–515.

St. John, M. F. & McClelland, J. (1990), 'Learning and applying contextual constraints in sentence comprehension', *Artificial Intelligence* **46**, 5–46.

Stolcke, A. (1990), Learning feature-based semantics with simple recurrent networks, Technical Report TR-90-015, International Computer Science Institute, Berkeley, California.

Sun, G., Giles, C. L., Chen, H. & Lee, Y. (1993), The neural network pushdown automaton: Model, stack and learning simulations, Technical Report UMIACS-TR-93-77, Institute for Advanced Computer Studies, University of Maryland, College Park, MD.

Touretzky, D. S. (1989*a*), Rules and maps in connectionist symbol processing, Technical Report CMU-CS-89-158, Carnegie Mellon University: Department of Computer Science, Pittsburgh, PA.

Touretzky, D. S. (1989*b*), Towards a connectionist phonology: The "many maps" approach to sequence manipulation, *in* 'Proceedings of the 11th Annual Conference of the Cognitive Science Society', pp. 188–195.

Watrous, R. & Kuhn, G. (1992), 'Induction of finite-state languages using second-order recurrent networks', *Neural Computation* **4**(3), 406.

Williams, R. & Zipser, D. (1989), 'A learning algorithm for continually running fully recurrent neural networks', *Neural Computation* **1**(2), 270–280.

Williams, R. & Zipser, D. (1990), Gradient-based learning algorithms for recurrent connectionist networks, *in* Y. Chauvin & D. Rumelhart, eds, 'Backpropagation: Theory, Architectures, and Applications', Erlbaum, Hillsdale, NJ.

Zeng, Z., Goodman, R. & Smyth, P. (1994), 'Discrete recurrent neural networks for grammatical inference', *IEEE Transactions on Neural Networks* **5**(2), 320–330.