On the Applicability of Neural Network and Machine Learning Methodologies to Natural Language Processing

Steve Lawrence*, C. Lee Giles†, Sandiway Fong
lawrence@elec.uq.oz.au, {giles,sandiway}@research.nj.nec.com

NEC Research Institute
4 Independence Way
Princeton, NJ 08540

Technical Report
UMIACS-TR-95-64 and CS-TR-3479
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

Abstract

We examine the inductive inference of a complex grammar - specifically, we consider the task of training a model 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, Fransconi-Gori-Soda and Back-Tsoi locally recurrent networks, Elman, Narendra & Parthasarathy, and Williams & Zipser recurrent networks, Euclidean and edit-distance nearest-neighbors, simulated annealing, and decision trees. The feed-forward neural networks and non-neural network machine learning models are included primarily for comparison. We address the question: How can a neural network, with its distributed nature and gradient descent based iterative calculations, possess linguistic capability which is traditionally handled with symbolic computation and recursive processes? Initial simulations with all models were only partially successful by using a large temporal window as input. Models trained in this fashion did not learn the grammar to a significant degree. Attempts at training recurrent networks with small temporal input windows failed until we implemented several techniques aimed at improving the convergence of the gradient descent training algorithms. We discuss the theory and present an empirical study of a variety of models and learning algorithms which highlights behaviour not present when attempting to learn a simpler grammar.

* Also with Electrical and Computer Engineering, University of Queensland, St. Lucia Qld 4072, Australia.
† Also with the Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

1 Motivation

1.1 Formal Grammars and Grammatical Inference

We give a brief introduction to formal grammars, grammatical inference, and natural language; for a thorough introduction we recommend Harrison [24] and Fu [15]. Briefly, a grammar G is a four tuple $\{N, T, P, S\}$, where N and T are sets of terminals and nonterminals comprising the alphabet of the grammar, P is a set of production rules, and S is the start symbol. For every grammar there exists a language L, a set of strings of the terminal symbols, that the grammar generates or recognizes. Grammatical inference is concerned mainly with the procedures that can be used to infer the syntactic or production rules of an unknown grammar G based on a finite set of strings I from $L(G)$, the language generated by G, and possibly also on a finite set of strings from the complement of $L(G)$ [15]. In this paper we consider replacing the inference algorithm with a neural network or a machine learning methodology. Our grammar is that of the English language. The simple grammar used by Elman [13] shown in figure 1 contains some of the structures in the complete English grammar: eg. agreement, verb argument structure, interactions with relative clauses, and recursion.

S	$\rightarrow$	NP VP "."
NP	$\rightarrow$	PropN | N | N RC
VP	$\rightarrow$	V (NP)
RC	$\rightarrow$	who NP VP | who VP (NP)
N	$\rightarrow$	boy | girl | cat...
PropN	$\rightarrow$	John | Mary
V	$\rightarrow$	chase | feed | see...

Figure 1: A simple grammar encompassing a subset of the English language.

In the Chomsky hierarchy of phrase structured grammars, the simplest grammar and its associated automata are regular grammars and finite-state-automata (FSA). However, it has been firmly established [4] that the syntactic structures of natural language cannot be parsimoniously described by regular languages. Certain phenomena (eg. center embedding) are more compactly described by context-free grammars which are recognised by push-down automata, while others (eg. crossed-serial dependencies and agreement) are better described by context-sensitive grammars which are recognised by linear bounded automata [42].

1.2 Representational Power

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] [40]. In the past few years several recurrent neural network architectures have emerged which have been used for grammatical inference [7] [20] [17] [18] [19]. Recurrent neural networks have been used for several smaller natural language problems, eg. papers using the Elman network for natural language tasks include: [50] [1] [11] [23] [32]. Neural network models have been shown to be able to account for a variety of phenomena in phonology [16] [21] [53] [52], morphology [22] [37] and role assignment [38] [32]. Induction of simpler grammars has been addressed often - eg. [54] [18] on learning Tomita languages [51]. Our task differs from these in that the grammar is considerably more complex.

It has been shown that recurrent networks have the representational power required for hierarchical solutions [13], and that they are Turing equivalent [47]. The recurrent neural networks investigated in this paper

[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 impractical for anything except relatively small grammars [40].

constitute complex, dynamical systems. Pollack [42] points out that Crutchfield and Young [8] 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 [33] [43].

1.3 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 [6]? 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 judgements are sometimes subtle but unarguably form part of the native speaker's language competence. In other cases, judgement 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 judgements 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 [5]) 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 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,

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

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 neural network or machine learning models 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 model from scratch, i.e. without the bifurcation into learned vs. innate components assumed by Chomsky, to produce the same judgements as native speakers on the sharply grammatical/ungrammatical pairs of the sort discussed in the next section.

2 Data

Our primary data consists of 552 English positive and negative examples taken from an introductory GB-linguistics textbook by Lasnik and Uriagereka [36]. Most of these examples are organized into minimal pairs like the example *I am eager for John to win/*I am eager John to win* that we have seen above. We note here that the minimal nature of the changes involved suggests that our dataset may represent an especially difficult task for the models. Due to the small sample size, the raw data, namely words, were first converted (using an existing parser) into the major syntactic categories assumed under GB-theory. Figure 2 summarizes the parts-of-speech that were used.

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* ...

Figure 2: Parts of Speech

The part-of-speech tagging represents the sole grammatical information supplied to the models 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 models.) 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 figure 3.

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

[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 ([41]).

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
I am eager to be here	n4 v2 a2 v2 adv	1
	n4 v2 a2 p1 v2 adv	1

Figure 3: Examples of Part-of-Speech Tagging

assigning the correct parts-of-speech, but no disambiguation was done to provide a greater challenge for the models. Furthermore, tagging resulted in several contradictory and duplicated sentences. Various methods were tested to deal with these cases, however we chose to remove them altogether for the results reported here. In addition, the number of positive and negative examples was equalised in all training and test sets in order to reduce any effects due to skewed data.

3 Data Encoding

For input to the models, the data was 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[4]. Two outputs were used in the neural networks, corresponding to grammatical and ungrammatical classifications. A confidence criteria was used: $y_{max} \times (y_{max} - y_{min})$[5].

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 performed simulations using a number of different neighborhood sizes ranging from 1 to 20. The best performance was obtained using a neighborhood size of only one. 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

[4] A fixed length window made up of segments containing 23 separate inputs, corresponding to the classifications noun class 1, noun class 2, verb class 1, etc.. was also tested but proved inferior.

[5] For an output range of 0 to 1.

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 [6].

We were unable to attain greater than 55% correct classification on average. 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[7].

5 Decision Tree Methods

Decision tree methods construct a tree which partitions the data at each level in the tree based on a particular feature of the data. CLS [29] used a heuristic lookahead method to construct decision trees. ID3 [44] extended CLS by using information content in the heuristic function. We tested the C4.5 algorithm by Ross Quinlan [45], which is an industrial strength version of ID3 designed to handle noise.

C4.5 only deals with strings of constant length and we used an input space corresponding to the longest string - we do not expect C4.5 to be highly suitable to the problem. The default C4.5 parameters were used. We expect that significantly more data would be required to match the performance of recurrent neural networks due to the position dependence created by the fixed input window. We obtained 60% correct classification performance on the test data on average.

6 Neural Network Models

A brief description of the models tested follows:

1. *Multi-layer perceptron.* The output of a neuron is computed using[8]

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

2. *Frasconi-Gori-Soda locally recurrent networks.* The Fransconi-Gori-Soda network has a locally recurrent globally feedforward architecture which includes a feedback connection around each hidden layer node [14]. Fransconi-Gori-Soda define local-output and local-activation versions of the architecture where the feedback is taken from the respective points. We have used the local-output version where the output of a node, $y(t) = f(wy(t-1) + \sum_{i=0}^{n} w_i x_i)$.

3. *Back-Tsoi FIR [2].* An FIR filter and gain term is included in every synapse.

[6]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 [34].

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

[7]Consider how you would define the cost for deleting a noun without knowing the context in which it appears.

[8]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.

Definition 1 An FIR MLP with L layers excluding the input layer $(0, 1, ..., L)$, FIR filters of order n_b, and $N_0, N_1, ..., N_L$ neurons per layer, is defined as

$$y_k^l(t) = f\left(x_k^l(t)\right) \tag{2}$$

$$x_k^l(t) = \sum_{i=0}^{N_{l-1}} c_{ki}^l(t) \sum_{j=0}^{n_b} w_{kij}^l(t) y_i^{l-1}(t-j) \tag{3}$$

where $y(t)$ = neuron output, c_{ki}^l = synaptic gain, $f(\alpha) = \frac{e^{\alpha/2} - e^{-\alpha/2}}{e^{\alpha/2} + e^{-\alpha/2}}$, $k = 1, 2, ..., N_l$(neuron index), $l = 0, 1, ..., L$(layer), and $y_k^l|_{k=0} = 1$(bias). □

4. *Narendra and Parthasarathy.* A feed-forward network augmented with feedback connections from the output nodes to the hidden nodes. As described in [39].
5. *Elman.* A simple recurrent network with feedback from each hidden node to all hidden nodes as described in [12], [13].
6. *Williams and Zipser.* A fully recurrent network where all non-input nodes are connected to all other nodes as described in [55].

We expect the feedforward and locally recurrent architectures to encounter difficulty performing the task and include them primarily as control cases.

7 Gradient Descent Learning

We have used backpropagation through time[9] [56] to train the globally recurrent networks[10], standard backpropagation for the multi-layer perceptron, and the gradient descent algorithms described by the authors for the locally recurrent networks. The error surface of a multilayer network is non-convex, non-quadratic, and often has large dimensionality. We found the standard gradient descent algorithms to be impractical for our problem[11]. We investigated the techniques described below for improving convergence. Except where noted, the results in this section are for networks using: two word inputs for the Elman network (the current and previous word) and single word inputs for the Williams & Zipser network (in order to keep the dimensionality roughly equal), 10 hidden nodes, the quadratic cost function, the logistic sigmoid function, sigmoid output activations, one hidden layer, the learning rate schedule shown below, an initial learning rate of 0.2, the weight initialisation strategy discussed below, and one million stochastic updates. Due to the dependence on the initial parameters, we have attempted to make the results as significant as possible by performing multiple simulations with different initial weights and training set/test set combinations. However, due to the complexity of the task[12], we could not perform as many simulations as desired. The standard deviation of the NMSE values are included to help assess the significance of the results.

Normalised mean squared error results are defined as

[9] 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.

[10] Real-time [55] recurrent learning was also tested but did not show any significant convergence for our problem.

[11] Without modifying the standard gradient descent algorithms we were only able to train networks which operated on a large temporal input window. These networks were not forced to model a grammar, they only memorised and interpolated between the training data.

[12] Each individual simulation in this section took an average of two hours to complete on a Sun Sparc 10 server.

Definition 2 NMSE $= \frac{\sum_{k=1}^{N}(d(k)-y(k))^2}{\left(\sum_{k=1}^{N}\left(d(k)-\left(\sum_{k=1}^{N}d(k)\right)/N\right)^2\right)/N}$. The denominator is the variance of the target values. k ranges over all patterns and outputs in the dataset. □

1. *Detection of Significant Error Increases.* If the NMSE increases significantly during training then 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.* Targets 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 update attempts to follow the true gradient, whereas stochastic is similar to adding noise to the true gradient. There are advantages and disadvantages of each method as shown in table 1.

Method:	Stochastic update	Batch update
Advantages	1. Faster convergence - update after every pattern 2. Stochastic trajectory helps avoid local minima	1. Guaranteed convergence to (local) minima 2. Second order update techniques available
Disadvantages	1. Most second order methods perform poorly 2. NMSE jumps around, less robust parameter updating 3. Convergence proofs probabilistic 4. Harder to parallelize	1. Very slow on large problems 2. Increased tendency to converge to poor local minima

Table 1: Comparison of stochastic and batch updating.

Table 2 shows the average NMSE and standard deviation using four different simulations for each case. The training times were equalised by reducing the number of updates for the batch case. Batch update often converges quicker using a higher learning rate than the optimal rate used for stochastic update[13], hence we investigated altering the learning rate for the batch case. We were unable to obtain significant convergence.

4. *Weight initialisation.* Random weights are initialised with the goal of ensuring that the sigmoids do not start out in saturation but are not very small (corresponding to a flat part of the error surface) [25]). In addition, several sets of random weights are tested and the set which provides the best performance on the training data is chosen[14]. In our experiments on the current problem, we have found that these techniques do not make a significant difference as shown in table 3 where the results are averaged over four simulations.

[13] Stochastic update does not generally tolerate as high a learning rate as batch update due to the noise injected by the stochastic nature of the updates.

[14] The number of weight sets tested is normally proportional to the number of weights squared, however we set a limit of twenty for these simulations.

	Stochastic		Batch	
	NMSE	Std. Dev.	NMSE	Std. Dev.
Elman	0.366	0.035	0.931	0.0036

Table 2: Training set NMSE comparison for batch versus stochastic update.

	Single random weights		Multiple weights	
	NMSE	Std. Dev.	NMSE	Std. Dev.
Elman	0.376	0.038	0.369	0.036

Table 3: Training set NMSE comparison for single or multiple initial random weights.

5. *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 [9] [10]:

$$\eta(t) = \frac{\eta_0}{1 + \frac{t}{\tau}} \tag{4}$$

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[15] and hence we have added an additional term to further reduce the learning rate over the final epochs (our specific learning rate schedule can be found in a later section).

Table 4 compares the performance of four averaged simulations for the two cases: using a constant learning rate and using a learning rate schedule. We have found the use of learning rate schedules to improve performance considerably.

	Constant		Schedule	
	NMSE	Std. Dev.	NMSE	Std. Dev.
Elman	0.742	0.154	0.394	0.035

Table 4: Training set NMSE comparison for a constant learning rate and a learning rate schedule.

6. *Sigmoid functions.* Symmetric sigmoid functions (eg. tanh) often improve convergence over the standard logistic function. For our particular problem we found that the difference was minor and that the logistic function resulted in better performance. Table 5 shows the averaged results of four simulations for each case.

7. *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. This enabled the networks to focus on the simpler data first. Elman suggests

[15] 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.

Sigmoid:	Logistic		Tanh	
	NMSE	Std. Dev.	NMSE	Std. Dev.
Elman	0.387	0.023	0.405	0.14
W & Z	0.650	0.022	0.835	0.13

Table 5: Training set NMSE comparison for logistic and tanh sigmoid activation functions.

that the initial training constrains later training in a useful way [13]. Results of four simulations per case comparing the use of sectioning with standard training on our problem are shown in table 6. The use of sectioning has consistently decreased performance. Why do we obtain the opposite results to those obtained by Elman? We suggest that in our case initial training constrains later training in a negative fashion: initial constraints may make modelling grammatical constructs only found in later data more difficult.

	No Sectioning		Sectioning	
	NMSE	Std. Dev.	NMSE	Std. Dev.
Elman	0.367	0.011	0.573	0.051
W & Z	0.594	0.084	0.617	0.026

Table 6: Training set NMSE comparison for the use of training data sectioning.

8. *Cost function.* The relative entropy cost function has received particular attention ([3] [27] [49] [25] [26]) and has a natural interpretation in terms of learning probabilities [35]. We investigated using both quadratic and relative entropy cost functions:

Definition 3 The quadratic cost function is defined as $E = \frac{1}{2}\sum_k (y_k - d_k)^2$ □

Definition 4 The relative entropy cost function is defined as $E = \sum_k \left[\frac{1}{2}(1+y_k)log\frac{1+y_k}{1+d_k} + \frac{1}{2}(1-y_k)log\frac{1-y_k}{1-d_k}\right]$ □

where y and d correspond to the actual and desired output values, k ranges over the outputs (and also the patterns for batch update).

Table 7 shows the results of four simulations for each case. 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 range of weight changes and therefore decreased robustness in parameter updating.

Sigmoid:	Quadratic		Entropy	
	NMSE	Std. Dev.	NMSE	Std. Dev.
Elman	0.470	0.078	0.651	0.0046
W & Z	0.657	0.019	0.760	0.023

Table 7: Training set NMSE comparison for logistic and tanh sigmoid activation functions.

8 Simulated Annealing

Previous work has shown the use of simulated annealing for finding the parameters of a recurrent network model to improve performance [48]. For comparison with the gradient descent based algorithms we have investigated using simulated annealing to train exactly the same Elman network as has been successfully trained to 100% correct training set classification using backpropagation through time (details in a later section). We have obtained no significant results from these trials[16]. Currently, the best trial has only obtained an NMSE of 1.2 after two days of execution on a Sun Sparc 10[17]. In comparison, the successful Elman models obtain an NMSE of approximately 0.1. We have not found the use of simulated annealing to improve performance, as Simard et. al. [48] have. Their problem was the parity problem with only four hidden units.

9 Results

Our results are based on multiple training/test set partitions and multiple random seeds. We have also used a set of Japanese control data. Japanese is at the opposite end of the language spectrum when compared with English and we expect a model trained on the English data to perform poorly on the Japanese data. Indeed, all models do perform poorly on the Japanese data.

Results of initial simulations using all models are shown in table 8. Using a large temporal input window, the neural network models (except the Williams and Zipser model) 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 recurrent network exceeded 55% correct classification on the test data. The Elman network was able to attain 74% correct classification on the test data. This is better than the performance obtained using any of the other networks, however it is still quite low. The available data is quite sparse and we expect increased generalisation performance as the amount of data increases, as well as increased difficulty in training.

The best neural network models were then selected from the full set for more in-depth analysis. Five simulations were performed for each network architecture. Each simulation took approximately four hours on a Sun Sparc 10 server. Table 9 summarises the results obtained with the various networks where each network contained 20 hidden nodes. Table 10 summarises the results for simulations where each network contained 30 hidden nodes. In order to make the number of weights in each architecture approximately equal we have used only single word inputs for the Williams & Zipser model but two word inputs for the others. Experiments indicate that this reduction in dimensionality for the W & Z network was desirable.

The results for the 30 hidden units case are generally worse than those for the 20 hidden units case but follow a similar trend across architectures. We also ran simulations with 10 hidden units where the trend across architectures was again similar but the performances were lower again (the Elman network attains roughly 88% correct classification on the training data and 65% on the English test data).

9.1 Network Operation

In this section we take a closer look at the operation of the networks analysed in detail. The error during training for a sample of each network architecture is shown in figure 4. The errors shown in the graph are the

[16] We have used the adaptive simulated annealing code by Lester Ingber [30] [31].

[17] The package used for simulated annealing by Lester Ingber includes speedups to the basic algorithm but even when we used these the algorithm did not converge.

TRAIN	large window	small window
MLP	100	55
FGS	100	56
BT-FIR	100	56
Elman	100	**100**
W&Z	94	92

TEST	large window	small window
Edit-distance	55	-
Euclidean	65	55
Decision trees	60	-
MLP	63	54
FGS	65	55
BT-FIR	64	54
Elman	65	**74**
W&Z	62	71

Table 8: Average results summary.

TRAIN	Classification	Std. dev.	Confidence
Elman	99.6%	0.84	78.8%
FGS	67.1%	1.22	17.6%
N & P	75.2%	1.41	32.2%
W & Z	91.7%	2.26	63.2%

ENGLISH TEST	Classification	Std. dev.	Confidence
Elman	74.2%	3.82	75.4%
FGS	59.0%	1.52	18.8%
N & P	60.6%	0.97	26.9%
W & Z	71.3%	0.75	65.1%

JAPANESE TEST	Classification	Std. dev.	Confidence
Elman	45.5%	2.0	87.2%
FGS	56.7%	12.2	18.4%
N & P	57.2%	10.6	23.9%
W & Z	50.0%	12.5	62.6%

Table 9: Results of the network architecture comparison for networks containing 20 hidden units.

NMSE over the complete training set and that parameters are being updated throughout each epoch. Note the nature of the Williams & Zipser learning curve and the utility of detecting and correcting for significant error increases[18].

Figure 5 shows an approximation of the "complexity" of the error surface based on the first derivatives of

[18]The learning curve for the Williams & Zipser network can be made smoother by reducing the learning rate but this tends to promote convergence to poorer local minima.

TRAIN	Classification	Std. dev.	Confidence
Elman	98.3%	1.2	72.1%
FGS	62.8%	0.39	11.2%
N & P	62.5%	0.23	13.8%
W & Z	93.7%	1.0	73.4%

ENGLISH TEST	Classification	Std. dev.	Confidence
Elman	72.1%	6.2	65.6%
FGS	57.6%	0.20	11.9%
N & P	55.8%	0.63	12.9%
W & Z	69.3%	5.0	69.9%

JAPANESE TEST	Classification	Std. dev.	Confidence
Elman	45%	11.5	67.3%
FGS	32%	0.22	11.4%
N & P	38%	10.3	20.4%
W & Z	59%	13.5	70.5%

Table 10: Results of the network architecture comparison for networks containing 30 hidden units.

the error criterion with respect to each weight $C = \frac{\sum_i \frac{\partial E}{\partial w_i}}{N_w}$ where i sums over all weights in the network and N_w is the total number of weights. This value has been plotted after each epoch during training. Note the more complex nature of the plot for the Williams & Zipser network.

Figure 6 shows the individual partial derivatives of the error criterion with respect to each weight after training has completed. Note the varying ranges of the values and the generally higher values for the Williams & Zipser network.

These plots give an insight into the nature of the error surface for each architecture. The Williams & Zipser fully connected network is more powerful than the Elman architecture, yet the Williams & Zipser error surface is far more complex - leading to difficulty for gradient descent optimization, and decreased performance. The relative simplicity of the error surface for the less powerful networks is evident.

Figure 7 shows a sample error surface with respect to the two most sensitive weights (largest $\frac{\partial E}{\partial w_i}$ values) for a sample run of each architecture. While these graphs vary considerably from trial to trial, the samples shown have been observed to be typical. These graphs are of minimal value compared to the true error surface but back up observations on complexity from the previous graphs.

9.2 Complete Simulation Details

Complete details on a sample Elman network follows:

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 were within the range zero to one. All target outputs were either 0.1 or 0.9. Bias inputs were used. The best of 20 random weight sets was chosen based on training set performance. Weights were initialised as shown in Haykin [25]. The logistic output activation function was used. The quadratic cost function was used. The search then converge learning rate schedule used was $\eta = \frac{\eta_0}{\frac{n}{N/2} + \frac{c_1}{max\left(1,(c_1 - \frac{max(0,c_1(n-c_2N))}{(1-c_2)N}\right)}}$ where η = learning rate, η_0 = initial learning rate, N = total training

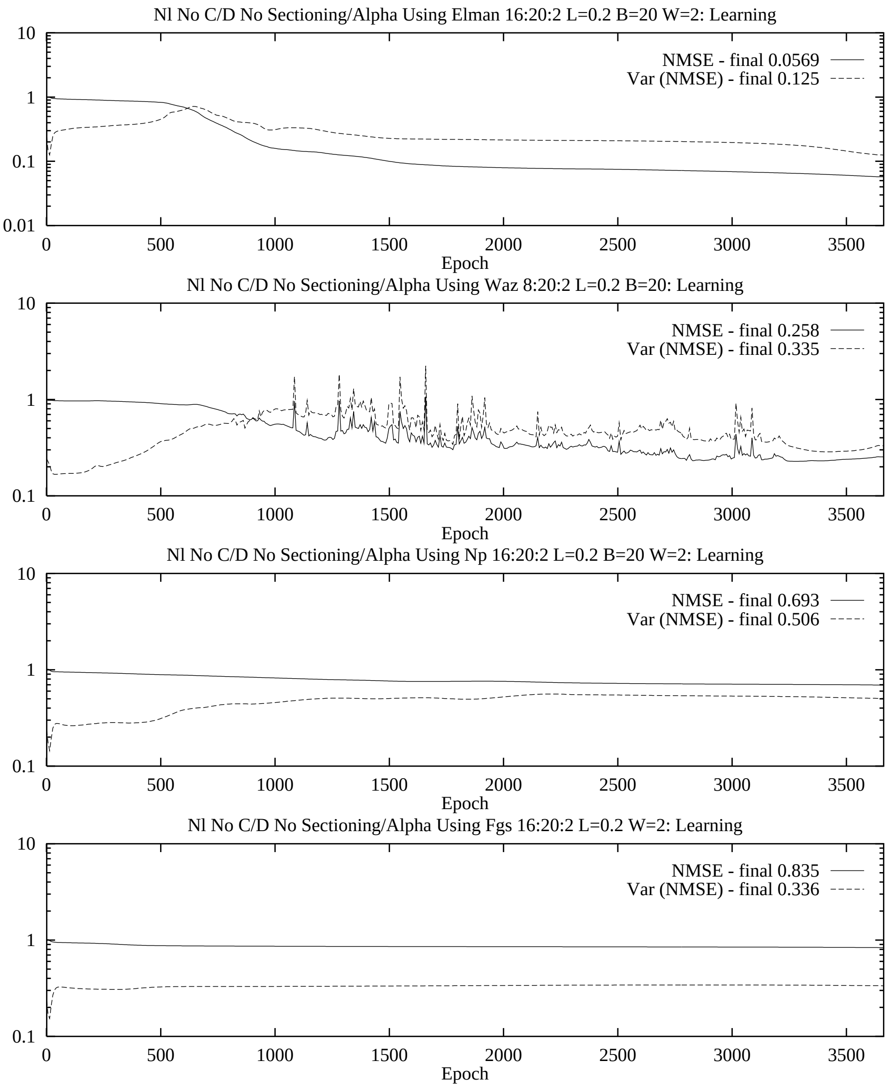

Figure 4: Average NMSE over the training set and variance of the NMSE over the training set during training. Top to bottom: Elman, Williams & Zipser, Narendra & Parthasarathy and Fransconi-Gori-Soda.

epochs, n = current training epoch, $c_1 = 50$, $c_2 = 0.65$. The training set consisted of 373 non-contradictory examples as described earlier. The English test set consisted of 100 non-contradictory samples and the Japanese test set consisted of 119 non-contradictory samples. 73% correct classification was obtained on the English test set and 53% correct classification was obtained on the Japanese test set. Figures 8 to 12 show the NMSE and the learning rate during training, along with training and test set confidence histograms for the cases where the output was correct and where the output was incorrect.

10 Discussion

Are the methods really learning anything? Nearest-neighbors, decision trees, and feed-forward neural networks do not learn a 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

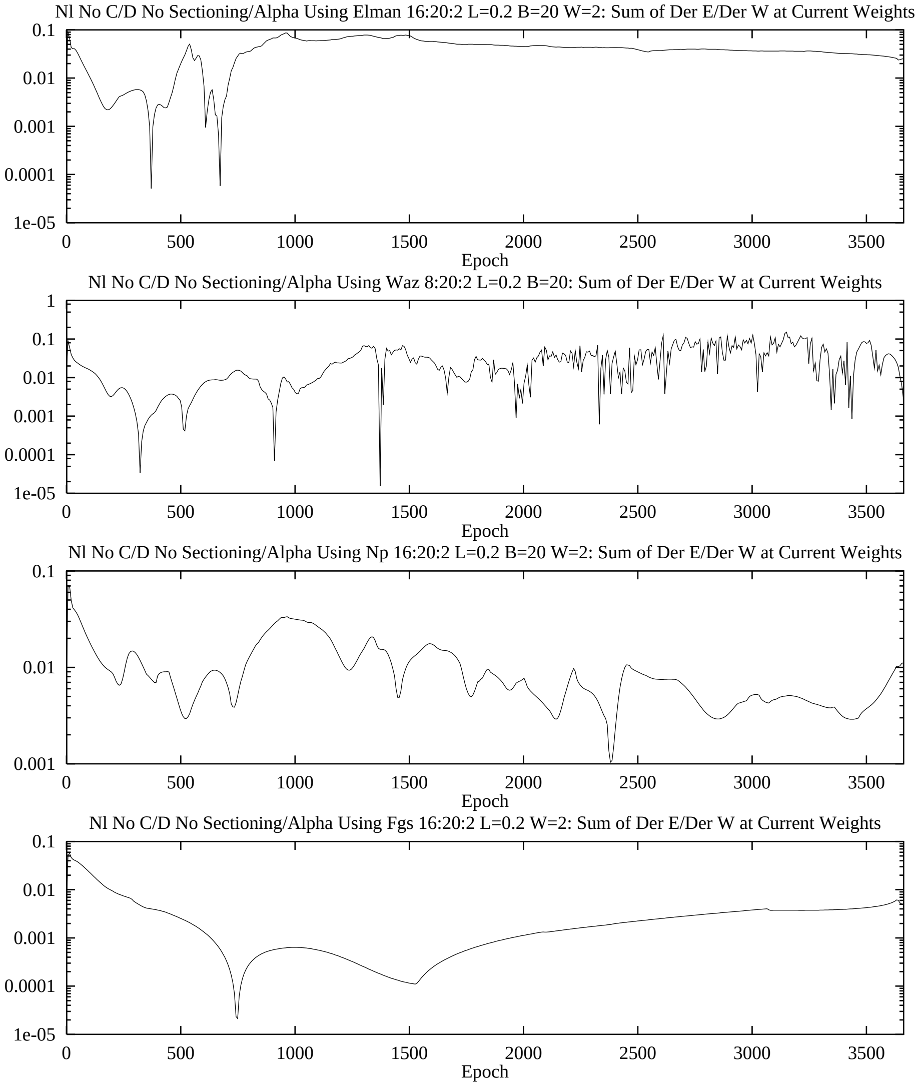

Figure 5: Approximate "complexity" of the error surface during training. Top to bottom: Elman, Williams & Zipser, Narendra & Parthasarathy and Fransconi-Gori-Soda.

only a small temporal input window without forming a state machine.

We investigated the use of Fransconi-Gori-Soda, Narendra & Parthasarathy, Elman and Williams & Zipser networks in detail. From best to worst performance, the architectures are: Elman, Williams & Zipser, Narendra & Parthasarathy and Fransconi-Gori-Soda. Theoretically, the Williams & Zipser network is the most powerful in terms of representational ability, yet the Elman network provides better performance. Investigation shows that this is due to the more complex error surface of the Williams & Zipser architecture. This result is supported by the parsimony principle [46]. Results indicate that a global minimum is never found for the task and algorithms described here, however, we note that the local minima which are found consistently possess performance which is similar within each architecture.

The hierarchy of architectures with increasing computational power give an insight into whether or not the increased power is used to model the more complex structures found in the grammar. The Fransconi-Gori-Soda network can be considered a control case - it is not capable of representing the kind of structures found in natural language. Its poor performance gives us more confidence that the modelling power of

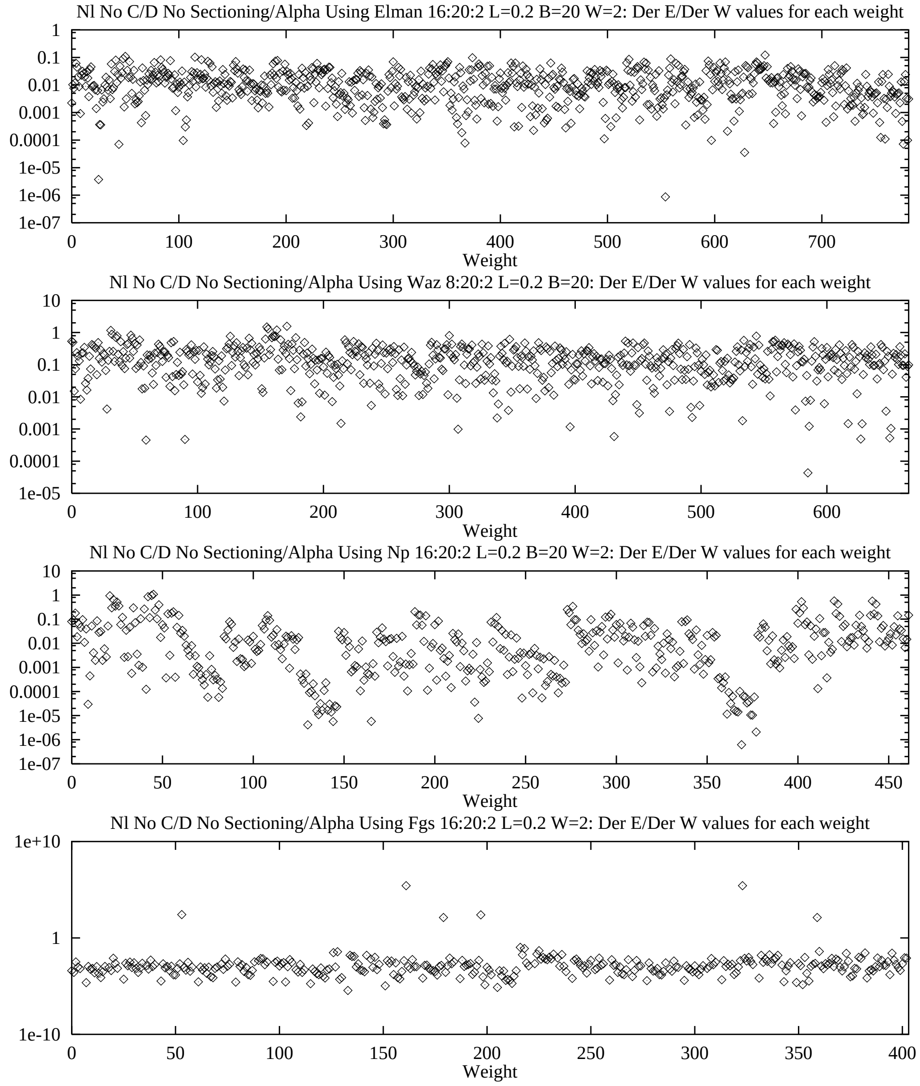

Figure 6: $\frac{\partial E}{\partial w_i}$ for each weight in the networks. Top to bottom: Elman, Williams & Zipser, Narendra & Parthasarathy and Fransconi-Gori-Soda.

the remaining architectures is being utilised by the learning algorithms. The fact that the more powerful Elman and Williams & Zipser networks do provide increased performance suggests that they are able to find structure in the data which cannot be modelled by the remaining networks. Additionally, analysis of the data suggests that 100% correct classification on the training data with only two word inputs would not be possible without learning significant aspects of the grammar.

Another comparison of recurrent neural network architectures, that of Giles and Horne [28], compared various networks on randomly generated 6 and 64-state finite memory machines. The locally recurrent and Narendra & Parthasarathy networks proved as good or superior to more powerful networks like the Elman network, indicating that either the task did not require the increased power, or the vanilla backpropagation through time learning algorithm used was unable to exploit it.

Although we have demonstrated the ability of the Elman and Williams & Zipser networks to model the grammar, we found considerable difficulty in finding suitable parameters for the models. Standard gradient descent methods did not work. Experimenting with various techniques for improving the convergence of

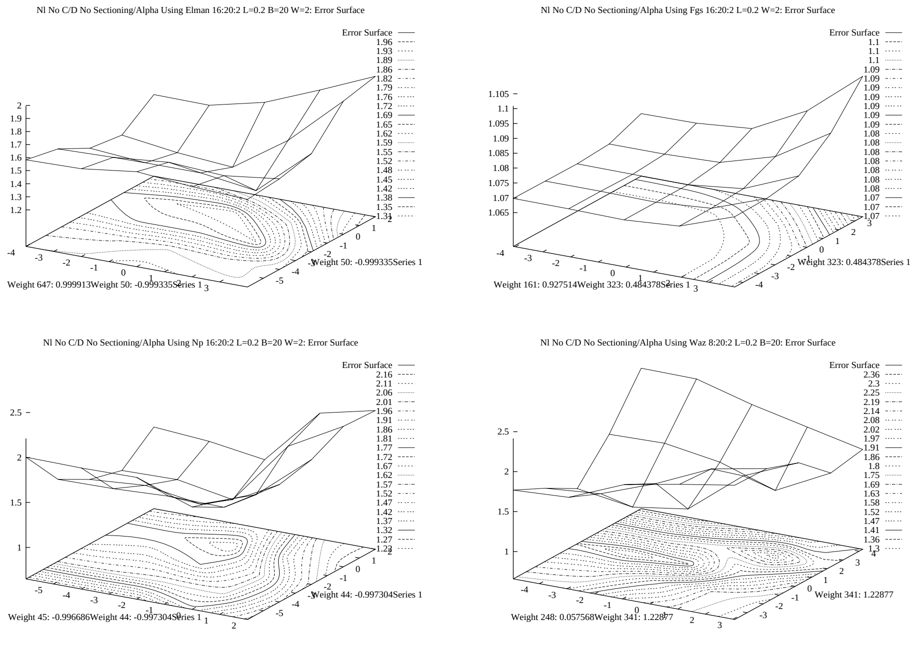

Figure 7: Error surface with respect to the two most sensitive weights. Clockwise from top-left: Elman, Fransconi-Gori-Soda, Williams & Zipser and Narendra & Parthasarathy.

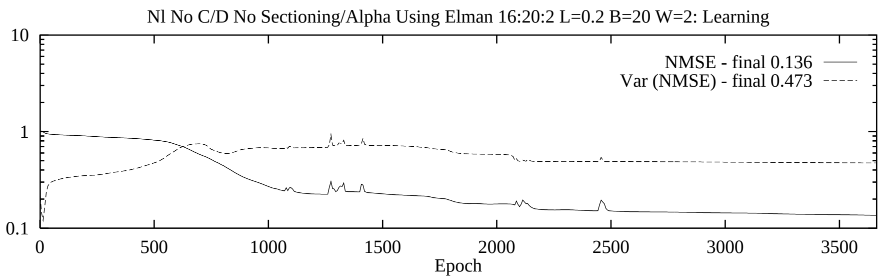

Figure 8: Average NMSE over the training set and variance of the NMSE over the training set during training.

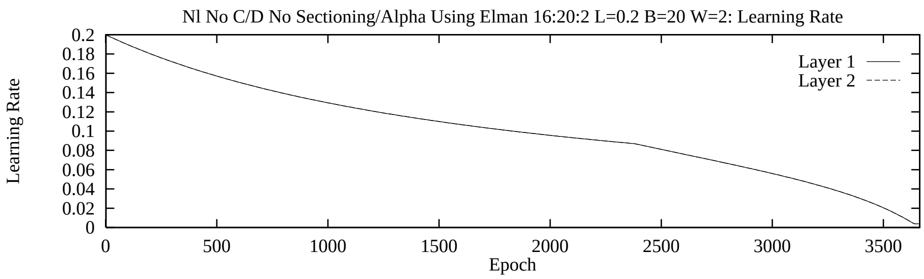

Figure 9: The learning rate during training.

standard gradient descent eventually led to success, however, this is not ideal: some techniques add extra parameters to the process, and analysis of convergence is difficult. We investigated using simulated annealing for finding the parameters - a technique with much stronger theoretical convergence proofs. However, we found the technique to be too computationally expensive.

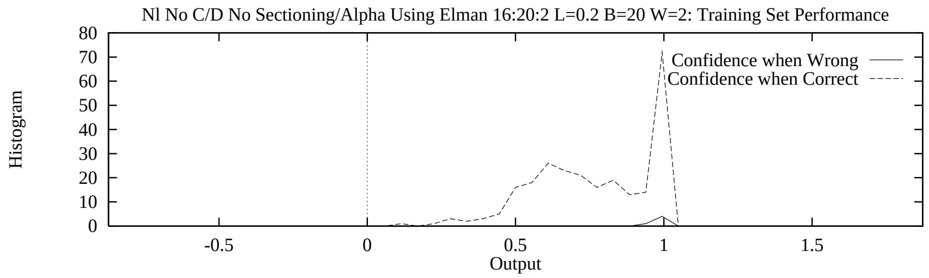

Figure 10: Training set performance.

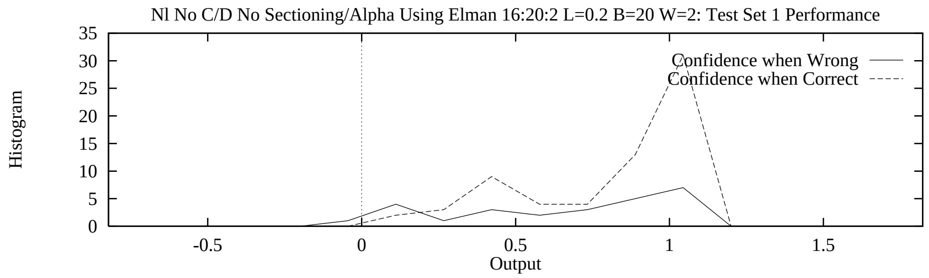

Figure 11: English test set performance.

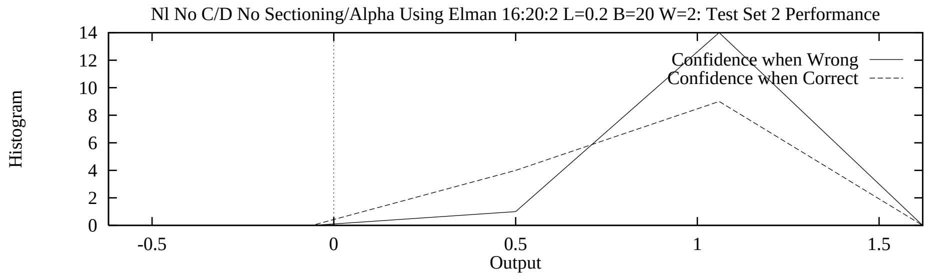

Figure 12: Japanese test set performance.

We cannot make significant conclusions based on the generalisation of the models due to the sparseness of the available data, however we believe that our comparison of architectures and algorithms on a complex problem and success in training the Elman and Williams & Zipser networks is significant. With an increase in the amount of data we expect better generalization performance, however we also expect training to be more difficult. It is clear that there is considerable difficulty scaling the models considered here up to larger problems. Hence, we need to continue to address the convergence of the training algorithms. We believe that further improvement is possible by continuing to address the nature of parameter updating during gradient descent. However, a point must be reached after which improvement with gradient descent based models requires consideration of the nature of the error surface. Further insight may be gained by investigating: the input and output encodings (these are not commonly chosen with the specific aim of controlling the error surface), the ability of parameter updates to modify network behaviour without destroying previously learned information, and the method by which the networks implement structures such as hierarchical and recursive relations.

References

[1] Robert B. Allen. Sequential connectionist networks for answering simple questions about a microworld. In *5th Annual Proceedings of the Cognitive Science Society*, pages 489–495, 1983.

[2] A.D. Back and A.C. Tsoi. FIR and IIR synapses, a new neural network architecture for time series modelling. *Neural Computation*, 3(3):337–350, 1991.

[3] E.B. Baum and F. Wilczek. Supervised learning of probability distributions by neural networks. In D.Z. Anderson, editor, *Neural Information Processing Systems*, pages 52–61, New York, 1988. (Denver 1987), American Institute of Physics.

[4] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, IT-2:113–124, 1956.

[5] N.A. Chomsky. *Lectures on Government and Binding*. Foris Publications, 1981.

[6] N.A. Chomsky. *Knowledge of Language: Its Nature, Origin, and Use*. Prager, 1986.

[7] A. Cleeremans, D. Servan-Schreiber, and J. McClelland. Finite state automata and simple recurrent recurrent networks. *Neural Computation*, 1(3):372–381, 1989.

[8] J. P. Crutchfield and K. Young. Computation at the onset of chaos. In W. Zurek, editor, *Complexity, Entropy and the Physics of Information*. Addison-Wesley, Reading, MA, 1989.

[9] Christian Darken and John Moody. Note on learning rate schedules for stochastic optimization. In *Neural Information Processing Systems 3*, pages 832–838. Morgan Kaufmann, 1991.

[10] Christian Darken and John Moody. Towards faster stochastic gradient search. In *Neural Information Processing Systems 4*, pages 1009–1016. Morgan Kaufmann, 1992.

[11] Jeffrey L. Elman. Structured representations and connectionist models. In *6th Annual Proceedings of the Cognitive Science Society*, pages 17–25, 1984.

[12] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

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

[14] P. Frasconi, M. Gori, M. Maggini, and G. Soda. Unified integration of explicit rules and learning by example in recurrent networks. *IEEE Transactions on Knowledge and Data Engineering*, 1992. To appear.

[15] K.S. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliffs, N.J, 1982.

[16] M. Gasser and C. Lee. Networks that learn phonology. Technical report, Computer Science Department, Indiana University, 1990.

[17] C.L. Giles, D. Chen, C.B. Miller, H.H. Chen, G.Z. Sun, and Y.C. Lee. Second-order recurrent neural networks for grammatical inference. In *1991 IEEE INNS International Joint Conference on Neural Networks - Seattle*, volume II, pages 273–281, Piscataway, NJ, 1991. IEEE Press.

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

[19] C.L. Giles, C.B. Miller, D. Chen, G.Z. Sun, H.H. Chen, and Y.C. Lee. Extracting and learning an unknown grammar with recurrent neural networks. In J.E. Moody, S.J. Hanson, and R.P Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 317–324, San Mateo, CA, 1992. Morgan Kaufmann Publishers.

[20] C.L. Giles, G.Z. Sun, H.H. Chen, Y.C. Lee, and D. Chen. Higher order recurrent networks & grammatical inference. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 380–387, San Mateo, CA, 1990. Morgan Kaufmann Publishers.

[21] M. Hare. The role of similarity in hungarian vowel harmony: A connectionist account. Technical Report CRL Tech report 9004, Centre for Research in Language, University of California, San Diego, 1990.

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

[23] Catherine L. Harris and Jeffrey L. Elman. Representing variable information with simple recurrent networks. In *6th Annual Proceedings of the Cognitive Science Society*, pages 635–642, 1984.

[24] M.H. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1978.

[25] S. Haykin. *Neural Networks, A Comprehensive Foundation.* Macmillan, New York, NY, 1994.

[26] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley Publishing Company, Inc., Redwood City, CA, 1991.

[27] J.J. Hopfield. Learning algorithms and probability distributions in feed-forward and feed-back networks. *Proceedings of the National Academy of Sciences, USA*, 84:8429–8433, 1987.

[28] B. G. Horne and C. Lee Giles. An experimental comparison of recurrent neural networks. In *Advances in Neural Information Processing Systems 7*, page to appear, 1995.

[29] E.B. Hunt, J.Marin, and P.T.Stone. *Experiments in Induction.* Academic Press, New York, NY, 1966.

[30] L. Ingber. Very fast simulated re-annealing. *Mathl. Comput. Modelling*, 12:967–973, 1989.

[31] L. Ingber. Adaptive simulated annealing (asa). Technical report, Lester Ingber Research, McLean, VA, ftp.caltech.edu: /pub/ingber/asa.Z, 1993.

[32] M. F. St. John and J. L. McLelland. Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence*, 46:5–46, 1990.

[33] A. K. Joshi. Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions? In L. Karttunen D. R. Dowty and A. M. Zwicky, editors, *Natural Language Parsing*. Cambridge University Press, Cambridge, 1985.

[34] Joseph B. Kruskal. An overview of sequence comparison. In David Sankoff and Joseph B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, Massachusetts, 1983.

[35] S. Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

[36] H. Lasnik and J. Uriagereka. *A Course in GB Syntax: Lectures on Binding and Empty Categories.* MIT Press, 1988.

[37] B. MacWhinney, J. Leinbach, R. Taraban, and J. McDonald. Language learning: cues or rules? *Journal of Memory and Language*, 28:255–277, 1989.

[38] R. Miikkulainen and M. Dyer. Encoding input/output representations in connectionist cognitive systems. In G. E. Hinton D. S. Touretzky and T. J. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 188–195, Los Altos, CA, 1989. Morgan Kaufmann.

[39] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, 1(1):4, 1990.

[40] Fernando Pereira. Inside-outside reestimation from partially bracketed corpora. In *ACL 92*, 1992.

[41] D. M. Pesetsky. *Paths and Categories.* PhD thesis, MIT, 1982.

[42] J.B. Pollack. The induction of dynamical recognizers. *Machine Learning*, 7:227–252, 1991.

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

[44] J.R. Quinlan. Discovering rules from large collections of examples: a case study. In D. Michie, editor, *Expert Systems in the Microelectronic Age.* Edinburgh University Press, Edinburgh, 1979.

[45] Ross Quinlan. *C4.5 Programs for Machine Learning.* Morgan Kaufmann, San Mateo, California, 1993.

[46] J. Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific, Singapore, 1989.

[47] H.T. Siegelmann and E.D. Sontag. On the computational power of neural nets. In *Proceedings of the Fifth ACM Workshop on Computational Learning Theory*, pages 440–449, New York, N.Y., 1992. ACM.

[48] P.Y. Simard, M.B. Ottaway, and D.H. Ballard. Analysis of recurrent backpropagation. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 103–112, San Mateo, 1989. (Pittsburg 1988), Morgan Kaufmann.

[49] S.A. Solla, E. Levin, and M. Fleisher. Accelerated learning in layered neural networks. *Complex Systems*, 2:625–639, 1988.

[50] Andreas Stolcke. Learning feature-based semantics with simple recurrent networks. Technical Report TR-90-015, International Computer Science Institute, Berkeley, California, April 1990.

[51] M. Tomita. Dynamic construction of finite-state automata from examples using hill-climbing. In *Proceedings of the Fourth Annual Cognitive Science Conference*, pages 105–108, Ann Arbor, Mi, 1982.

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

[53] D. S. Touretzky. Towards a connectioninst phonology: The 'many maps' approach to sequence manipulation. In *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, pages 188–195, 1989.

[54] R.L. Watrous and G.M. Kuhn. Induction of finite-state languages using second-order recurrent networks. *Neural Computation*, 4(3):406, 1992.

[55] R.J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[56] R.J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent connectionist networks. In Y. Chauvin and D.E. Rumelhart, editors, *Backpropagation: Theory, Architectures, and Applications*. Erlbaum, Hillsdale, NJ, 1990.