The Limitations of Modeling Finite State Grammars with Simple Recurrent Networks

Erik F. Tjong Kim Sang
vakgroep Alfa-informatica
University of Groningen
e-mail: `erikt@let.rug.nl`

Abstract

In his book 'Mechanisms of Implicit Learning' (1993) Axel Cleeremans describes how finite state grammars can be modeled successfully with connectionist networks namely with Simple Recurrent Networks (SRNs) developed by Jeffrey Elman. However, SRNs cannot be used for modeling arbitrary finite state grammars. In this paper I describe the limitations of this approach.

1 Introduction

In my thesis research I am comparing the performance of three different machine learning techniques on the problem of acquiring models for the phonotactic structure of Dutch (Tjong Kim Sang 1996). A good phonological model is able to make the distinction between words that are in a language and words that cannot be part of the language because of their structure (for example *bda* in English). My goal is to find out what machine learning techniques are well suited for linguistic problems and I am also interested in the possible gain the learning processes can have from being supplied with initial knowledge.

One of the three learning techniques behaved surprisingly worse than could be expected from the results reported in the literature. The connectionist Simple Recurrent Networks (SRNs) were developed by Jeffrey Elman to be applied for generating or recognizing sequences (Elman 1990). In (Cleeremans et al. 1989), (Servan-Schreiber et al. 1991) and (Cleeremans 1993), Axel Cleeremans, David Servan-Schreiber and James McClelland report experiments in which they modeled finite state grammars with SRNs. The experiments were very successful: after a training phase in which the network only received positive examples it modeled the grammar perfectly, accepting all valid strings and rejecting all invalid strings.

The results reported in (Cleeremans 1993) motivated me to use SRNs for modeling the phonotactic structure of Dutch mono-syllabic words. Earlier approaches to this problem which used the statistical Hidden Markov Models and a rule-based learning technique have produced satisfactory results. Therefore we can safely assume that finite state grammars exist that are reasonable models for the phonotactic structure of Dutch mono-syllabic words. To my surprise SRNs failed in finding such a model. The SRNs reached the 100% acceptance rate for valid Dutch monosyllabic words but they never rejected more than 6% of the random strings.

In this paper I will explain why SRNs performed bad on my problem while their performance was excellent on the problem posed to them in (Cleeremans et al. 1989), (Servan-Schreiber et al. 1991) and (Cleeremans 1993). First I will describe the structure of an SRN and give a description of the main experiment described in the literature. After that I will present a summary of my own experiments and give an description of the problematic difference between my experiments and the experiment described in the literature. In the final part of the paper I will give two extensions of the grammar modeled by Cleeremans et al. and I will show that the performance of the SRNs degrades when they are required to learn grammars of increasing complexity.

2 The Simple Recurrent Network

Standard feed-forward connectionist networks (Rumelhart et al. 1986) consist of a sequence of layers of cells. Usually three layers of cells are present in the network: the input layer, the output layer and the in-between hidden layer (figure 1). Signals in these networks are represented by numbers between 0 and 1. The input layer cells receive signals from outside and send them to the hidden layer cells. They perform a computation on these signals and send the result to the output layer cells. These cells perform a computation on the signals and send their output to outside. This type of network is a pattern transformation system.

Networks of this type are capable of handling context insensitive pattern transformations ($A \rightarrow a$: replace A by a regardless of the context of A). However, they cannot perform context sensitive pattern transformations ($bA \rightarrow ba$: replace A by a only if A is preceded by b). For this purpose Jeffrey Elman developed the Simple Recurrent Network (SRN) (Elman 1990). This network is equal to a feed-forward network expanded with some context cells that are connected to the cells in the hidden layer with forward and backward connections (figure 1). These cells store the output signals of the cells of the hidden layer and output these signals at the next processing step. They give the network the memory that is necessary for performing context sensitive pattern transformations.

A typical task of an SRN concerns modeling a finite state grammar. Suppose we want to model the grammar:

S→aaS
S→b

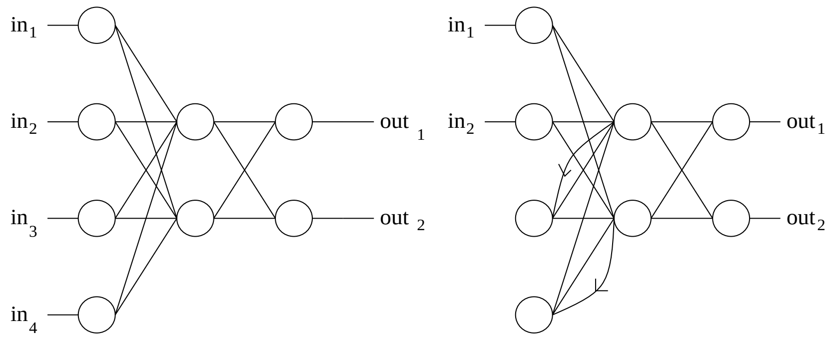

Figure 1: A standard feed-forward network (left) and a Simple Recurrent Network (right). Except when arrows indicate otherwise, all connections are forward connections.

This grammar can be modeled by the SRN shown in figure 1. Then a parse of a string consists of successively feeding the SRN the characters in the string and making it predict the next character. This task is non-deterministic: for a substring *aa* which is generated by our small grammar we cannot predict if the next character is *a* or if it is *b*. In the output of SRNs this is solved by giving all possible continuation tokens a value which is as high as possible.

Usually in an SRN each different token is assigned to a different cell. I define a maximum signal of 1 at cell in_1 as current character is *a* and signal of 1 at in_2 as current character is *b*. A maximum signal of 1 at out_1 means that the network predicts that the next character is *a* and signal of 1 at out_2 means that it predicts that the next character is *b*. The output of feed-forward networks and SRNs contains some noise. Furthermore, in this SRN set-up the sum of the output values is equal to 1 and this means that we cannot expect an output signal value larger than 0.5 if we have two valid continuations of a string. For these two reasons, I will accept that an output signal indicates a valid continuation character if the signal is larger than 0.3.

To make the SRN model the grammar, we will have to train it. I will not describe the training process here. Interested readers are referred to (Elman 1990) and (Cleeremans 1993). Suppose the SRN has been trained to model the grammar and that the training process terminated. Then a parse of the string *aab* will evolve as follows:

1. (*a*): start with in_1=1 and in_2=0. The output of the network should be $\text{out}_1 \geq 0.3$ and $\text{out}_2 < 0.3$. If that is the case, the network predicts that the next character is *a* which is correct.

2. (*a*): start with in_1=1 and in_2=0. The output of the network should be $\text{out}_1 \geq 0.3$ and $\text{out}_2 \geq 0.3$. If that is the case, the network predicts that the next character is *a* or *b* which is correct.

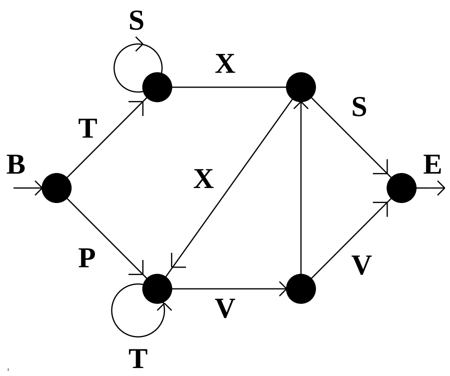

Figure 2: Finite state network representing the Reber grammar

3. (*b*): start with in_1=0 and in_2=1. The output of the network should be out_1 <0.3 and out_2 <0.3. If that is the case, the network predicts no next character which is correct.

The SRN accepts a string if at all processing steps the next character is among the possible continuation characters according to the output of the network.

3 The Experiments of Cleeremans et al.

In experiments described in (Cleeremans et al. 1989), (Servan-Schreiber et al. 1991) and (Cleeremans 1993), Axel Cleeremans, David Servan-Schreiber and James McClelland trained a network to recognize strings which were generated using a small grammar that was originally used by (Reber 1976) (figure 2). They trained an SRN to predict the next character in a sequence of 60,000 strings which were randomly generated by the grammar.[1] This prediction task is non-deterministic and the size of the network was too small to memorize the complete sequence. Therefore the network cannot perform this task without making errors. It was sufficient that the network indicated in its output what characters are valid continuations of the preceding sequence. In the experiments each possible character was represented by one cell in the input layer and one cell in the output layer. The aim of the experiments was to make the network output at least 0.3 in the cells that correspond to valid continuation characters. So if the output of the network was something like:

B	T	S	X	V	P	E
0.0	0.0	0.4	0.5	0.0	0.1	0.0

[1]Network training parameters: learning rate η and momentum α were in the range 0.01 to 0.02 and 0.5 to 0.9 respectively. The network contained seven input cells (BTSXVPE) , three hidden cells and seven output cells.

possible continuation tokens were S and X because these receive an output value higher than 0.3. If no character received a value of 0.3 the sequence was considered to be invalid.

Cleeremans, Servan-Schreiber and McClelland tested their network with 20,000 strings generated by the grammar represented by the finite state network shown in figure 2. For all characters in the strings the network output in the corresponding cells was 0.3 or higher. Thus the network accepted 100% of the strings. After this they fed the network 130,000 random strings built from the same characters. This time the network accepted 0.2% of the strings. It turned out that the accepted 0.2% of the strings were valid strings. All other strings were invalid according to the grammar. The network separated perfectly the grammatical strings from the non-grammatical strings.

4 Experiments with the Phonotactic Structure of Dutch

In my own experiments I have attempted to model the phonotactic structure of Dutch monosyllabic words with SRNs. The purpose of these experiments was to obtain an SRN that is able to recognize the difference between possible monosyllabic Dutch words like for example *bad* and strings that cannot be a monosyllabic word like for example *bda*. The words were represented in orthographic format (ordinary characters; no phonological representations) and were taken from the Dutch corpus *Het groene boekje* (Nederlands-Belgische Spellingscommissie 1954). From this corpus I obtained 3506 monosyllabic words. I divided this set in two corpora: one 3206-word training corpus (training data) and a 300-word test corpus (test data). As an additional test corpus I generated 300 random words (random data) of which 12 words (4.0%) were acceptable. During the training phase the SRN will be fed the positive training data only. After the training phase it has to accept as many of the test data as possible and reject as many random data as possible. In an experiment with the statistical learning technique Hidden Markov Models an acceptance rate of 99.3% (test data) and an rejection rate of 95.0% (random data) were achieved for the same data (Tjong Kim Sang 1996). While these were non-perfect scores, they indicate that reasonable models for the phonotactic structure of Dutch monosyllabic words *can* be found.

I chose the same network learning parameters as in the experiments of (Cleeremans 1993): learning rate η=0.01 and momentum α=0.5. The network contained 27 input cells (a-z$*$) and also 27 output cells. All words received a special word-end character $*$ because the words will be presented to the network in one large sequence and the network needs a word-end character in order to know where the current word ends and the next word starts. Determining the number of hidden cells of the SRN is a problem. This number cannot be derived analytically from other network parameters or from network input data. Yet the size of the hidden layer has some influence on the performance of the network. Therefore I performed three different experiments with hidden layer sizes of 4, 10 and 21 cells respectively.

Network training was carried out in steps of 50 training rounds. In each training round the SRN was presented the complete training data and the network was forced to adapt its internal weights in order to get its output as close to being perfect as possible. After a 50-rounds training period I examined the influence training had on the total sum squared error the network made for the training data. When the value of this error had changed more than 1% training was continued. I continued this process until the total sum squared error stayed within 1% distance of the previous value.

Before training the network weights were initialized with random values. I noticed that random initialization of the networks had some influence on their performance. In order to get a network performance that is reliable and stable, I performed five parallel experiments with each network configuration. Of these five experiments I chose the one with the lowest total sum squared error for the training data and tested its performance on the test and the random data. My motivation for choosing the performance of the network with the smallest error rather than the average performance of the networks is that I am interested in the best achievable performance of the network. This approach to learning to solve a problem is not uncommon.

Because there are more valid continuation tokens in the grammar I am looking for than in the Reber grammar, the output signals of my SRN will be smaller than the SRN modeling the Reber grammar. The output signal threshold of 0.3 used in (Cleeremans 1993) would reject almost all words. Therefore I changed the string score measure used in (Cleeremans 1993) into:

1. The score assigned by an SRN to a character X in a string Y is the value of the output cell which was assigned to X after processing the prefix of Y before X.

2. The score of a string is the lowest score assigned by the SRN to any of the characters in the string.

3. The STRING ACCEPTANCE THRESHOLD is equal to the lowest string score assigned by the SRN to a string of the training data.

So instead of determining a standard threshold value in advance, I have made the string acceptance threshold value depend on the scores of the training words. Strings in the test data and the random data will be rejected if they receive a lower score than the string acceptance threshold. All other strings will be accepted.
All SRNs performed perfectly for the test words and accepted all 300 words of the test data (figure 3). However, their performance for the random words was bad. The SRN with 4 hidden units got the highest rejection rate with 5.6% of the 300 random words. The network with 10 hidden units performed worse and the network with 21 hidden unit even accepted all random words.

hidden units	training rounds needed	total sum squared error	accepted test words	rejected random words
4	200	13750	300	17
10	250	13153	300	3
21	200	13060	300	0

Figure 3: The performance of three SRNs for the Dutch monosyllabic data.

The performance of the network is not anywhere near the 95-99% range of the Hidden Markov Model on this problem or the 100% performance of the SRN on the Reber grammar reported in (Cleeremans 1993).

5 The Influence of the Number of Possible Continuations of a String

My three experiments show that SRNs with the configuration I chose are unable to acquire a good model for the phonotactic structure of Dutch monosyllabic words. While my target was rejecting 96% of the random strings the SRNs never were able to reject more than 6%. This performance is a sharp contrast with the performance on grammaticality checking reported in (Cleeremans 1993) in which *all* non-grammatical strings were rejected by the network. This fact surprised me so I took a closer look at the differences between my experiments and the experiment described in chapter 2 of (Cleeremans 1993).

The most obvious difference I was able to find was the maximal number of valid continuations of grammatical strings. In finite state model for the Reber grammar used by Cleeremans et al. (figure 2) the maximum number of valid continuation tokens is two. This is very important for the format of the network output. This output consists of a list of real values, each roughly representing the chance that a certain token is the next one in the string. For example, if a network trained for the Reber grammar is processing a string, it might present on its output the pattern [0.00, 0.53, 0.00, 0.38, 0.01, 0.02, 0.00] (Cleeremans 1993, page 43). In this list the numbers indicate the probability that B, T, S, P, X, V, or E is the next token in the string. As we can see, the two valid continuations T (0.53) and P (0.38) receive an output value that is significantly larger than the output values of the other tokens. The network does not perfectly predict the 'correct' continuation but it outputs some average pattern that indicates possible continuation tokens. This network behavior was already recognized in (Elman 1990).

The fact is that in the phonotactic grammars for Dutch the maximal number of valid continuation tokens is much larger than two. For example, according to my training corpus for the plain text experiments the number of tokens which are possible continuations of the string s is 17. Some continuation tokens of s occur frequently, like c (23%) and t (24%), but others are very infrequent: f, q, u and w occur less than 1%. The frequency of the continuation tokens will be mirrored by the network output, as in the example from (Cleeremans 1993). Because the network output contains noise, it will be impossible to distinguish between low frequency continuation tokens and impossible continuation tokens, like b and x in this example.

In my experiments I chose the lowest character score in the training data as the string acceptance threshold value. By chance some impossible tokens will have received a character score which is larger than this threshold value. This results in the reported failure to reject all non-grammatical strings. Remember that the network output contains noise. For example, the output list in section 3 shows network output in which P receives value 0.1 (should be 0.0) and S receives value 0.4 (should be 0.5). Cleeremans et al. detected in the output of the network values of 0.3 where 0.5 was expected. One can understand that in an environment where signals have an absolute variation of 0.2 the difference between a signals 0.01 and 0.00 are in practice impossible to detect.

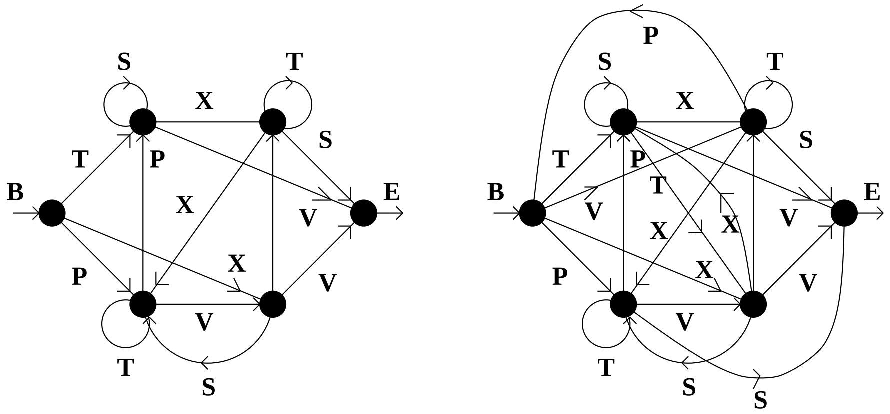

Figure 4: Finite state networks representing a Reber grammar with three continuation tokens (left) and one with four continuation tokens (right)

6 Can We Scale up the Cleeremans Experiment?

I decided to test the explanation I have given in the previous section by redoing the experiment described in chapter 2 of (Cleeremans 1993). Apart from training an SRN to decide on the grammatically of strings according to the Reber grammar shown in figure 2, I have trained SRNs with strings from two alternative grammars. The first alternative grammar was an extension of the Reber grammar such that valid substrings have 3 possible continuation tokens.

The second alternative grammar was an extension of the first and there valid substrings have 4 possible continuation tokens (figure 4). For all grammars I generated 3000 training words, 300 test words and 300 random words. Of the 300 random words for the standard Reber grammar 1 word was grammatical and for the other two grammars respectively 5 and 8 words were valid.

I trained the SRNs in steps of 50 training rounds using learning rate η=0.01, momentum α=0.5 and a network configuration that was similar to the one used in (Cleeremans 1993) (7 input cells, 3 hidden cells and 7 output cells). After each 50-round step I checked if the total sum squared error of the network had changed more than 1%. If this was the fact training was continued otherwise training was stopped. As in the previous experiments words were rejected when their score was below the string acceptance threshold, which was equal to the lowest score of a string in the training data.

maximum continuation tokens	training rounds needed	total sum squared error	rejected test words	rejected random words
2	150	9079	0	299
3	250	12665	0	272
4	100	13944	0	240

Figure 5: The performance of the SRN network for Reber grammars of increasing complexity.

As we can see in figure 5 the performance of the SRN decreases when the complexity of the grammar increases. The SRN accepted all valid words for all grammars but the rejection rate of the random data decreased when the number of valid continuation characters increased: for the grammar with 2 continuation characters (one valid string in the random data) 100% of the invalid random strings were rejected, for 3 continuation characters (5 valid strings) this dropped to 92.2% and for 4 continuation characters (8 valid strings) the rejection rate was 82.1%. We can conclude that the number of valid continuation tokens influences the difficulty of the problem. (Cleeremans et al. 1989), (Servan-Schreiber et al. 1991) and (Cleeremans 1993), showed that SRNs are capable of acquiring finite state grammars in which a grammatical substring has two valid continuation tokens. I showed that the performance of the SRNs will deteriorate rapidly if the maximal number of continuation tokens increases.

7 Concluding Remarks

In this paper I have described the limitations of modeling finite state grammars with the connectionist Simple Recurrent Networks (SRNs). This approach was promising after the successful experiments described in (Cleeremans 1993). However, the unsatisfactory results of my own experiments in which the phonotactic structure of mono-syllabic Dutch words was modeled with SRNs reduced this optimistic view. I have argued that the performance of the SRNs in the problem of the acquisition of a grammar depends on the number of valid continuations of a string. When there are more valid continuations possible, the probabilities of the individual continuations decrease and together with these the output values of SRN cells decrease. Lower cell output values are more difficult to distinguish from the inherent noise in the network output and therefore the SRN will have more difficulty modeling grammars with many possible valid string continuations. In the Reber grammar used by Cleeremans et al. a string had a maximum of two continuation characters while this number is 21 in a grammar describing the phonotactic structure of Dutch monosyllabic words. This accounts for the performance differences between these two experiments.

The problem described here is a fundamental problem of feed forward networks in this kind of experiment set-up. In the training data input-output dependencies are encountered infrequently and when there are many dependencies these are lost in the noise the network produces. This problem cannot be solved by changing the configuration or the size of the network. Still, theoretical research has proven that connectionists networks as basic as the McCulloch-Pitts networks can model finite state grammars (Kleene 1956). The SRNs I have used in my experiments should be able to model the finite state grammars shown in figure 4.

I believe that it is possible to train SRNs to model those grammars if the training data is adapted. Instead of supplying the network different individual relations one by one, it is necessary to supply them a group of relations: for example B$\rightarrow$ $[T, V, X, P]$ instead of B$\rightarrow$ T, B$\rightarrow$ V, B$\rightarrow$ X and B$\rightarrow$ P. This will able the network to reach output values for valid continuation characters that are close to 1. However, I also believe that this is an unsatisfactory adaptation of the data to the network. Therefore I would like to conclude that while SRNs are theoretically capable of modeling finite state grammars, in practice training them to simulate a non-trivial grammar is next to impossible.

References

Cleeremans, A. (1993). *Mechanisms of Implicit Learning.* The MIT Press.

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

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

Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. In Shannon, C. E., and McCarty, J., editors, *Automata Studies*, vol. 34 of *Annals of Mathematics Studies.* Princeton University Press.

Nederlands-Belgische Spellingscommissie (1954). *Woordenlijst der Nederlandse Taal.* Staatsdrukkerij. (Available by anonymous ftp from ftp://donau.et.tudelft.nl/pub/words/platte_lijst.Z).

Reber, A. S. (1976). Implicit learning of synthetic languages: The role of the instructional set. *Journal of Experimental Psychology: Human Learning and Memory*, 2.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, 318–362. MIT Press.

Servan-Schreiber, D., Cleeremans, A., and McClelland, J. L. (1991). Graded state machines: The representation of temporal continguincies in simple recurrent networks. *Machine Learning*, 161–193.

Tjong Kim Sang, E. F. (1996). *Machine Learning of Phonotactical Knowledge.* PhD thesis, University of Groningen. (to appear).