Modelling the Phonotactic Structure of Natural Language Words with Simple Recurrent Networks

Ivelin Stoianov, John Nerbonne and Huub Bouma

University of Groningen, Department of Humanities Computing, BCN

Abstract

Simple Recurrent Networks (SRN) are Neural Network (connectionist) models able to process natural language. Phonotactics concerns the order of symbols in words. We continued an earlier unsuccessful trial to model the phonotactics of Dutch words with SRNs. In order to overcome the previously reported obstacles, a new method for testing the network training was developed - *optimal threshold evaluation*. This method is based on minimising the erroneous character prediction of a trained SRN. The network training was improved as well. The training words were presented to the network according to their frequencies, which emphasizes the more frequent sequences. The achieved results are promising and provide a base for further study.

1 Introduction to connectionist natural language processing

It is still a challenge to process natural language with connectionist paradigms. Formal language theory provides more natural methods for exploring complex language phenomena, but if we search for an approach as robust and fast as the human brain is, we should take inspiration from the brain itself. Artificial neural networks (NNs) are computational models that resemble the brain structure and processing (Rumelhart et al., 1986; Elman, 1990; Dorffner, 1991). There are different NN models, but not all of them are capable of language processing. Some of them are designed to process static mappings, as Multilayered Perceptron (Rumelhart et al., 1986; Haykin, 1994), while other models are endowed with internal memory that enables them to process sequences. The latter NNs can be utilized for natural language processing (NLP) because language at all its levels — phonological, grammatical, etc. — has a sequential character which requires a mechanism in NNs that keeps information about the past — phonemes, words, etc.[1] Jeffrey Elman's (1990) Simple Recurrent Networks (SRN) and Plaut, McClelland, Seidenberg and Paterson's (1996) attractor networks belong to the second class, and they are successfully used for different linguistic tasks. The most influential and exploited model is the SRN (Miikkulainen and Dyer, 1991; Shillcock et al., 1993; Reilly, 1995; Stoianov,

[1] Static NNs can also process sequences, provided that the input has as many input nodes as the length of the sequence, times the size of a single input token representation. This extends the network size and processing time significantly.

1998). It comprises simplicity and representational power; therefore we selected this model for our experiments.

In spite of the growing number of studies related to connectionist NLP, there are still difficulties in the implementation of NNs for natural language processing. A problem recognized by most connectionist researchers is how to evaluate what the NN has learned during the training. Incorrect evaluation of training might lead even rigorous studies to a wrong interpretation of NN capacity, as in Tjong Kim Sang (1995). Therefore, in this study we explore this problem and propose a new evaluation method of SRN training, which we hope will contribute to our understanding of the correct application of NNs to linguistic tasks. Further, by learning the phonotactics of natural language words from a non-trivial wordlist, we will demonstrate the capacity of SRNs to solve complex linguistic problems.

The paper is organized as follows. First, in the following subsections, we provide background information about NNs and phonotactics, and we discuss related publications. Then we outline the data used. In the third section, we describe SRNs, the new evaluation method, and the results of the training. In the fourth one, we then provide an analysis of the learned phonotactics. Conclusions can be found in the last section. A detailed technical description of SRNs is given in appendix A. The second appendix contains a part of the training wordlist.

1.1 Neural Network premises

The human brain is built up from simple basic elements: neurons, which are organized in complex structures (Kandel, 1991). The speed and robustness of the brain are due to the parallel work of the neurons and the properties of this highly organized network. The brain is also adaptable: it learns and adjusts to its environment. The artificial brain analogues — NNs — are aimed firstly at explaining the brain itself and secondly at practical applications, which benefit from the advantages of brain-like computational models.

From a linguistic perspective, there are two main directions related to connectionist modelling. The first direction is explaining the linguistic capacity that humans possess. This problem is studied in cognitive science and, more specifically, in psycholinguistics and neurolinguistics. Some of the important problems are the functional organization of language processing brain structures (Hirshman and Jackson, 1997; Ainsworth-Darnell et al., 1998), their specific neurobiological localization (Stowe et al., 1994), associations to other modalities, etc. The other direction is connectionist natural language processing, including language parsing, generation and storage. The goal here is to compete with and even outperform existing methods, such as symbolic and stochastic approaches. In this area, the modelling of neural structures is compromized to obtain simpler, but computationally more effective models. SRNs belong to this category. Nevertheless, many authors use them to explain some characteristics of our linguistic capacity (Dell et al., 1993; Shillcock et al., 1993).

The NN structure is similar to the neuron assemblies in the brain. Knowledge and data in the NNs are represented distributively (Rumelhart et al., 1986;

Dorfner, 1991). The data is represented sparsely as the current activation of a set of neurons, usually organized in 'layers'. These activations are propagated from layer to layer. There is always an input layer, whose neurons are activated according to external environment, and an output layer, whose activations are the product of the network. The knowledge, or the rules of how to process the data, is redundantly encoded in the NNs as varying strengths of the connections between the neurons. Each 'rule' is represented by a large number of neurons and weights, and if small damage occurs, the rest of the neurons may be capable of producing a correct response. Moreover, they can learn quickly and can adjust their behaviour in a slightly modified configuration. This kind of representation is the basis for the generalization abilities of the NNs, i.e. the ability to generate similar output activity for similar input patterns even if they were unseen during the training. Most of the NN learning algorithms are also geared to this kind of distributed data representation. There are *unsupervised* learning algorithms that form 'rules' observing only input data, and *supervised* ones that may be told the correct response by a 'teacher' (a check on output). The presence of a teacher corresponds to learning an input/output mapping. One of the most popular supervised learning algorithms is Back-Propagation (BP) learning.

Biologically motivated NNs are supported further by a number of theoretical analyses. Hornik et al. (1991) proved that BP learning can approximate any continuous input/output static mapping to any degree of accuracy, by a multilayer neural network, if there are enough hidden neurons. Doya (1993) extended this result to include temporal patterns using recurrent neural architectures. The more restricted SRNs were proven by Sperduti (1997) to be able to simulate any frontier-to-root automaton, while some other recurrent models such as cascade correlation networks and Neural Trees can not. The latter kind of automaton recognizes tree grammars. The phonotactic structure of language words can easily be encoded in such a grammar, providing more reason to choose this recurrent NN model for our task.

1.2 Phonotactics

The principles of combination by which linguistic signs (symbols) form messages are called the grammar of the language. In the course of this research a phonological level of grammar is examined. Here, basic elements are phonemes. Phonotactic analysis considers the principles of their combination (Laver, 1994), i.e. the valid combinations of phonemes.

In the experiments we report on here, we used graphemes instead of phonemes, i.e. we studied combinations of letters. Using phonemes does not significantly increase the complexity of the problem: usually the phonetic representation is shorter than the orthographic representation, while the number of the phonemes is larger. Tjong Kim Sang (1998) also reports very little difference in the learning problems based on phonemes vs. letters. In spite of studying graphemes instead of phonemes, in the rest of the paper we will not use the more-specific term *graphotactics*, but *phonotactics*. The reader should keep in mind that we experiment with letters.

We will try to encode words from a natural language (Dutch) within a connectionist model, SRNs. The network will be given all words from a training wordlist. As a result, it should learn the phonotactics of these words. If the wordlist size is large enough, the NN will learn the phonotactics of the natural language the words belong to. Other approaches that might be used for studying phonotactics are N-grams and Hidden Markov Models.

Usually, the phonotactic rules are known in advance. An NN model trained to remember the phoneme combinations of a particular language should discover these rules. Therefore, one of the methods to examine whether this NN has learned the phonotactics of a given language is to check if these rules can be derived from the weight matrices of the NN. Another approach to examine the training is to verify how well the network recognizes strings, i.e. whether it can distinguish words belonging to the language from random strings. In this case, the dataset containing words from the training language has to be split into training and test sets. In addition, we make use of a random string set. After training, we will examine how many words from the test set are not recognized as belonging to the language, and how many words from the random string set are accepted. The average of these two errors will give an estimation of how well the NN recognizes the training language. This is the approach we implemented.

1.3 Related studies

There are few previous works that have attempted to learn phonotactics with NNs. The earliest report is by Shillcock et al. (1993). The attempt was to model the phonological space with SRNs. The network was trained to produce at the output layer the previous, current and future phoneme, given the current phoneme at the input. The phonemes were encoded in accordance with Government Phonology, in particular using feature-based representations. They reported unsuccessful prediction of the next phoneme. A follow-up paper by Cairns et al. (1997) considers phonotactics-based word segmentation and the reported results consider only segmentation.

Another connectionist model that learns to produce sequences of phonological features was presented by Dell (1993). Two recurrent network architectures were used: SRNs, and an extension of SRNs with an additional context layer receiving activation from the input layer. The purpose of this model was not phonotactic analysis, but producing the sequential phonetic representation of a static lexical term. Once a lexical term was presented to the input, the model activated the features of each segment of the input word, one segment at time. Experiments were done with a toy size dataset (50 words, each consisting of 3 phonemes) and with a larger dataset (300 – 400 words). Both architectures were reported to perform well, with superiority of the second model. The performance was measured as the number of correctly predicted phonemes. The reported error rate, depending on the network topology and training dataset, is 5 –10 %.

An attempt to scale up the problem is reported by Tjong Kim Sang (1995). He considered exactly the same problem we study; moreover it inspired our research. Success was achieved using Hidden Markov Models, but not using SRNs. It was concluded that the very complex structure of a grammar, describing a large number of words (3500 monosyllabic Dutch words) is not straightforward for SRNs. The reason for this was conjectured to be the large number of possible continuations. We propose a method to overcome this problem. The wordlist Tjong Kim Sang used is a part of the monosyllabic wordlist we use. He reported experiments with Hidden Markov Models as well, with classification error rate of 3%, which in turn can be suggested as a base-line error rate for stochastic models. SRNs are also a kind of probabilistic machine.

Another report considering connectionist learning of phonetic regularities in Turkish was published a few months after our first successful experiments (Rodd, 1997). The NN model was SRN, but the goal was different: finding similar phonemes formed in the hidden layers during the training. The training set was smaller, and no information about the training and performance was given.

Some other connectionist approaches applied to natural language learning — Lawrence et al. (1995; 1996), Elman (1990), Cleeremans et al. (1989), and Servan-Schreiber et al. (1991) among others — primarily considered syntax and some toy-sized problems. As pioneering projects in connectionist language processing, most of them were generally aimed at comparison between different approaches (including NNs) to language learning. But usually the attempts at scaling to larger problems were not successful.

2 Data and encoding

A wordlist containing all 4480 monosyllabic Dutch words was extracted from the CELEX lexical database.[2] The mean word length was 5 characters. A small part of this list can be found in Appendix 2. The CELEX database contains a few hundred thousand words from English, Dutch and German. This database is very suitable for linguistic research, because it has a lot of specific information related to each word; for example, the number of the syllables in the words, their frequencies, grammatical information, etc. We used the orthographic word representation. As we explained above, the complexity of studying graphemes and phonemes is similar. Also, in Dutch, unlike English, there is close similarity between orthographic and phonetic representation.

The extracted wordlist was split into a training set (80%) and a test set (20%). In order to test the discrimination capabilities of a trained SRN, a further random string set was generated. The length of the random strings and the symbol distribution in each of these followed the empirical distribution of the word length and symbol distribution in the training set. This improved the quality of the test.

[2] The CELEX lexical database is distributed on CD-ROMs. For more information see http://www.kun.nl/celex/ or Centre for Lexical Information, Nijmegen, The Netherlands.

The words were presented to the network letter by letter. At the end of each word, one extra symbol '#' ('end of word') was provided. In total, there were 27 symbols: 'a' ... 'z', '#'. All characters were orthogonally encoded in a vector of size 27. We selected orthogonal representation because it does not implicitly encode any significant relations among sounds or letters, which the learning should be required to uncover. Also, any non-orthogonal random representation can be transformed to this orthogonal one, and this can be computed with an extra non-recurrent hidden layer in the NN. The vectors are encoded as 0.1 at all positions except the one that stands for the particular character number, which is set to 0.9. Using 0.1 instead of 0 and 0.9 instead of 1 increases the effectiveness of the training process.

An important characteristic of a word is its frequency, i.e. how often the word is used. We constructed the training set to reflect frequency: words used more frequently appear on the training list more frequently. This emphasizes the more frequent symbol sequences. As a result, the network will be trained to respond better to more frequent words and if it is used to spell-check a Dutch text, for instance, it will make fewer errors in total. Still, the difference in the frequencies is too large. Certain words, such as *de* (the), occur millions of times more often than some rare words. Therefore, the word frequencies used during training depend logarithmically on the actual word frequencies. This reduces the range of appearance to a maximum of 100 times in one training epoch.

3 SRNs and phonotactics learning

The basic structure of SRNs as originally described by Elman is given in Figure 1, which also depicts the mechanism of sequence processing. The SRNs were developed as an extension of the popular feedforward supervised NN model, Multilayered Perceptron (Rumelhart, 1986), and they use the same basic computations. The only difference is the recurrent connection from the hidden layer to the 'context' layer, which can be seen as a part of the input layer, but fed with internally produced information. This context layer is used to remember the activation of the hidden layer at the previous moment and it is the temporal memory of SRNs. So, SRN response at each moment depends both on the current input and on the contextual memory. During the training process, SRNs build up an internal representation of the sequential input data in the context layer.

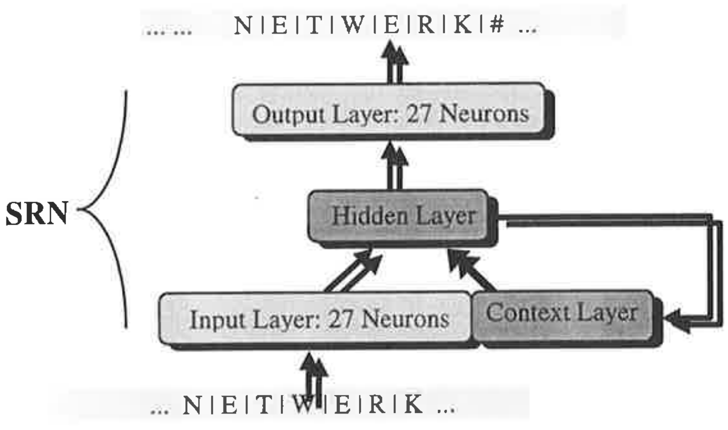


Figure 1: An SRN and the mechanism of sequence processing. One character is presented as input to the SRN, and the next one is used for training. In turn, it has to be predicted during the test phase.

The network learns the phonotactic rules by studying the possible successors after each left context in the training set. The training process is organized in epochs. In each epoch, all words are presented to the network. The frequency of a word's appearance in an epoch is proportional to (the logarithm of) its frequency in the language. Before starting a new word, the context is reset. A word is presented to the SRN letter by letter. The letter following the current input letter is the current successor. This is what the network should predict and should produce at the output. A training error is defined as the difference between the desired and the current output activation for each neuron. This error is used by the learning algorithm that adjusts the weights, so that the next time the same input is presented, the output is closer to the desired output.

SRNs can be trained with a standard BP algorithm or using Back-Propagation Through Time (BPTT) learning. Both of them allow the network to settle to a good point in the weight space, but the second one is faster and allows the network to adjust more precisely to the dependencies between context, current input and target letters. The difference is that the weights receive delta- or error-signals not only from the errors in the previous layer, but also from the propagated error in the context units. Therefore, we recommend that BPTT be used. In Appendix A, we provide technical details related to SRNs and BPTT implementation.

Since the task usually is not deterministic, i.e. there are many possible successors after a given left context, the output error can never reach zero (except where left contexts are unique in the training set). Therefore, as a result of the training, the network tends to learn the distribution of the allowable successors, i.e. the probability of each letter to follow the current context.

3.1 Evaluation of the training — optimal threshold

Learning phonotactics means learning symbol order. To check how well a trained SRN has learned the phonotactics of the training language, we can test its prediction performance. One possible method is to interpret the NN output as the likelihood that any symbol follows in the current context. That is, for each left context available in the language, to compute the proximity between the network response and the empirical distribution of all symbols, potential successors. This evaluation method is explored in a related paper by Stoianov (1998). Another method, explored by Cleeremans et al. (1989) and Tjong Kim Sang (1995) among others, is based on the idea that if the network accepts (almost) all valid sequences and rejects (almost) all invalid ones, then it has learned the phonotactics. We will pursue this approach further.

Now, how should we decide whether the joint response of the output layer $\{on(t_0), on(t_0+1), ..., on(t_0+L-1)\}$ to a string $[c_1\ c_2\ ..\ c_L]$ indicates that it is a part of the training language, or not? The first ideas put forward were the following: Cleeremans's implementation accepts a sequence if the responses of all designated neurons standing for the expected symbols in the sequence have an activation level greater than 0.3. Recall that in the orthogonal representation, there will be exactly one 'designated' neuron that represents a given symbol. Tjong Kim Sang's SRNs accept a string if the designated neurons are more active than the minimal activation value encountered in all training words. The problem with the first approach is that it is very task-specific. The value 0.3 is experimentally observed: in Cleeremans's task, there are no more than two successors and 0.3 is the threshold that optimally splits grammatical from non-grammatical strings. Applying this method to a task where the possible successors are 27 would fail, because only few of the neurons would have activation larger than 0.3. Tjong Kim Sang faced this problem and applied the second rule, but this turned out to accept not only the correct words, but almost all random sequences as well.

The solution we propose and use to good effect uses the same principle as the previous approaches: the activation of the neuron $on(t_0+i-1)$ standing for the expected symbol $\mathbf{c}_i$ is compared with a threshold τ. If for all of the relevant neurons, this activation is greater than or equal to τ, the string is accepted as being close enough to the words from the training set, i.e. as phonotactically like the words learned by the SRN during training. In this case, τ should be between zero and one. In the above notations, the index i stands for the position of the letter $\mathbf{c}_i$ in the tested word, i.e. it has temporal meaning. Since all phonemes from the tested word have to be confirmed by the NN, we can term this rule 'all-or-nothing'. Now, the question is how to set this threshold to distinguish grammatical from ungrammatical input sequences as effectively as possible. After a complete training, or after each training session, when we want to evaluate the training so far, we can examine the response of the net to all genuine words and to some randomly generated strings, for different threshold values. The network training is supposed to adjust the weights so as to improve the response for words from the training set, and to minimize the error in the test set. We test how many words would be accepted and rejected for different values for the threshold, and select the threshold that minimizes the combined error involving acceptance of random strings and rejection of genuine words. This use of negative data (random strings) departs from Cleeremans's and Tjong Kim Sang's work – Cleeremans uses negative data only to evaluate the SRN learning and Tjong Kim Sang uses only positive data to find the threshold (which is far from the optimal one).

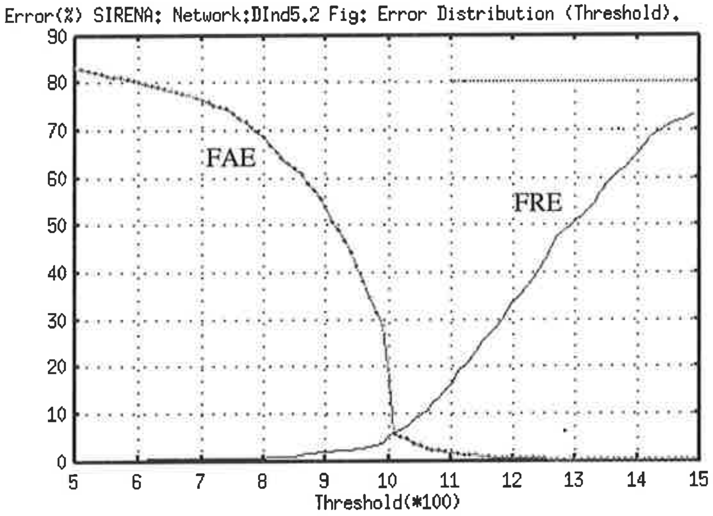


Figure 2. Falsely rejected word error (FRE) and falsely accepted random string error (FAE). The errors are given against the threshold τ (0.05...0.15).

Figure 2 depicts both the *falsely rejected word error* (FRE) and the *falsely accepted random strings error* (FAE), where the x-axis stands for threshold and the y-axis represents the error depending on the threshold. We can see that as the threshold increases, FRE increases too, because more words are rejected, while on the other hand FAE decreases, because fewer random strings are accepted. The point where the two errors are minimal is the threshold that is most interesting. This minimal threshold will be used for character acceptance. The corresponding error at this point is also used for evaluating the training.

The method being proposed here is independent of the number of successors, so it can work with 26 characters, as well with, for instance, 100 phonemes.

3.2 Details of the experiments and results

The model described so far was implemented in C++ for HP-Unix. We experimented with both BP and BPTT learning algorithms. BP required fewer computations and was faster and easier for implementation, but the training was longer. This is because BPTT propagates back through time temporal dependencies and uses them for training, while BP uses only the context layer as source for information for the past.

In the course of experiments, we fixed the following SRN parameters: learning rate $\eta = 0.3$ and network momentum $\alpha = 0.5$. These values were found to be optimal for various tasks. We used one hidden layer, which was copied to the context layer using the weighting coefficient $\beta = 0.8$. The only parameter we allowed to vary was the number of hidden nodes, from 5 to 30.

The training was organised in epochs, during which each word from the training set was selected in random order and used for training. The probability of a word being selected was equal to its frequency. This allowed the more frequent words and phoneme combinations to be studied more often than the infrequent ones.

After one training epoch, we evaluated the training by presenting all words from the training set to the NN and computing the error as a function of the threshold τ. The error function (in percent) depended also on the word frequency freq(word$_i$):

$$\mathrm{Err}\,(\tau) = 100\ |\mathrm{Words}|^{-1} \sum_{i=1\,..\,|\mathrm{Words}|} \mathrm{Err}_\tau(\mathrm{word}_i) * \mathrm{freq}(\mathrm{word}_i)$$

where |Words| is the number of the words in the training set and $\mathrm{Err}_\tau(\mathrm{word}_i)$ is the error for word *i* at a threshold τ. The same error function was computed as well for a random string set. Note that the error $\mathrm{Err}_\tau(\mathrm{word}_i)$ in the latter case is not the rejection of a random string, but its acceptance as a word belonging to the training language. The frequency of each random string was assessed as one. Finally, as we described in section 3.1, the threshold τ at which the combined error is minimal was found and used to evaluate the training.

The number of the training epochs depended on the epoch error behaviour. The training was terminated either if the error dropped below some threshold or failed to decrease significantly. On average, 100 to 300 epochs were enough to reduce the training error satisfactorily. After the training, a control test with the test set was performed, in order to test the generalisation abilities of the network. Usually, the test error was close to the training error.

A typical error shape after the training is shown in Figure 2. It was quite surprising for us that a SRN with as few as 5 to 10 hidden neurons managed to obtain about 7% error. Some other complex tasks, for instance in vision processing, require many more hidden neurons (more than 100). Although this performance is much better than the acceptance reported in Tjong Kim Sang (1995) of all testing strings and rejection of only a few random strings (95% error rate), we hope optimizations of the learning will improve the results even further.

4 Static analysis of the learned phonotactics

In this section we present a brief analysis of the phonotactics that a SRN with five hidden neurons has learned. The analysis will be based on two static representations of the weight matrixes of a trained SRN: a Hinton diagram and cluster analysis. The extracted rules are compared to corresponding phonotactic transition rules in Dutch, documented in Cohen et al. (1972).[3]

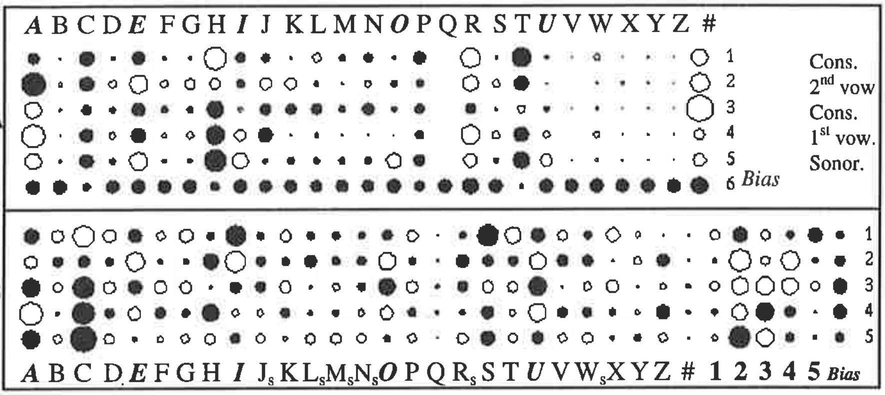
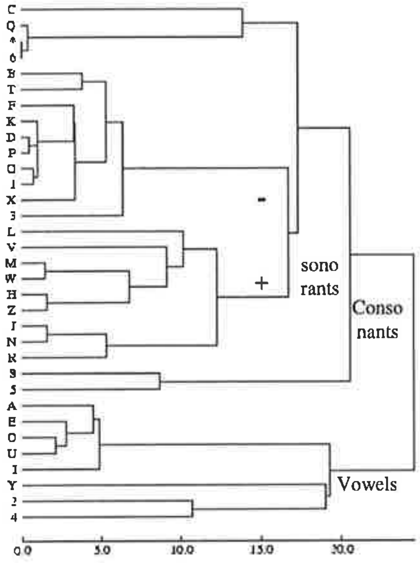


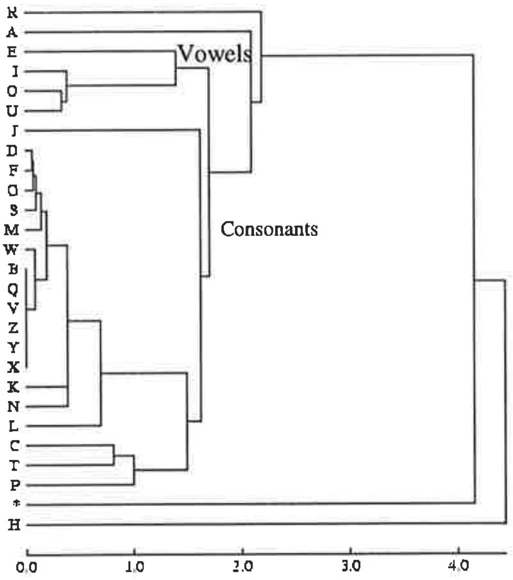
Figure 3. Hinton diagram of trained SRN weights: (a) Hidden to Output Layer; each column represents one output neuron. (b) Input Layer (A...Z,#) and Context Layer (1...5) to Hidden Layer; each row represents one hidden neuron. Note, e.g., that positive weights (empty circles) on the 2nd and 4th neuron (row 2,4) correspond to vowels.

The basic source of analysis is a Hinton diagram of the weight matrices for both hidden to output layer and input (including the context neurons) to hidden layers (Figure 3). In this diagram, each circle represents one weight. The strength and the sign of this weight are represented by the size and the colour of the circle. A white circle stands for positive and a black one stands for negative weights. Recalling that a neuronal output is a function of the product inputs - weights (see formulae (1) and (4) in Appendix 1), we might extract some simple rules.[4]

[3] We did not find a proper source with graphotactic rules, so the resemblance can only be approximate.
[4] We cannot draw exact rules because there are recurrent connections and biases that complicate the computations.



(a)



(b)

Figure 4. Cluster analysis of the Input characters (a) and Output characters (b) with respect to the weights connecting Input-Hidden and Output-Hidden layers. The numbers in (a) represent the context neurons. What we have labelled 'sonorants' in (a) includes voiced continuants (as well as sonorants proper).

The same weights, but analysed with cluster analysis and represented with dendograms are given in Figure 4. The left dendogram classifies each input letter with respect to the influence on the activation of the hidden neurons (hN). The right dendogram classifies the output letters with respect to the weights on its connections from the hidden layer.

First, we can see in Figure 4 that the weight vectors (and correspondingly the characters) cluster into known natural classes — vowels and consonants — in both input and output character clustering. The 'back' vowels (O,U) are subclustered from the 'front' (I,E,A) vowels. The consonants also are distinguished in the input character cluster regarding sonority. We will stop here searching for further known phonetic clusters, because the symbols are not phonemes, but graphemes. In spite of that, since these classes are formed only experiencing the words, we can conclude that the possible combinations in natural languages depend on the (phonetic) characteristics of the symbols.

As a further step, we will look at the possible transitions between two neighbouring symbols in Figure 3 and how they relate to the transition rules in a monosyllabic Dutch word (Cohen, 1972), which is of the following form:

$$\sigma \{\text{onset } [[[C^f]C^f]C^f] \text{ nucleus } V[V] \text{ coda } [C^b\ [C^b\ [C^b\ [C^b]]]]\ \}$$

where C stands for consonants and V for vowels. The general principle is that we can see what can follow what by tracing from (b) into (a) in Figure 3 over activated neurons.

According to Figure 3b, consonants and some context neurons activate the 1st, 3rd and 5th hNs. At the same time, the 2nd and 4th hNs are activated by vowels and context neurons. Now, if we look at Figure 3a, we see that the 2nd hN activates consonants and *end-of-word*, while the 4th hN activates vowels and consonants. Our interpretation of this is that the 2nd hN stands for an optional second vowel, while the 4th hN stands for the first vowel in the nucleus. Also, this should mean that one vowel can not end a syllable, while two vowels can. Now, if we turn to the phonotactics, there is a rule in Dutch that a short vowel cannot end syllables, only a long vowel. In monosyllabic words, the short vowels are represented mostly by a single letter, while the long vowels have two-character orthographic representations. We can see that the two rules match nicely.

With regard to the consonants, note that according to Dutch phonotactics, there cannot be an obstruent that intervenes between a sonorant in the onset or coda and the nucleus (syllable peak). What the SRN found is exactly the same: the sonorants (J,L,M,N,R,W) activate only the 5th hN, which in turn activates only vowels, 'R' and *end-of-word*. As for the obstruents, they activate the 1st and 3rd hNs at the input layer (Figure 3b), which in turn allows consonants and vowels to be activated at the output layer (Figure 3a). Another rule that can be found in Dutch is that the last consonant in a [CCCC]-coda can only be 'T'. This phenomenon is reflected in Figure 3a, where the only consonant activated by the 3rd hidden neuron is 'T', and this neuron itself is activated only by consonants, and context neurons including itself.

Finally, we would like to stress again that the rules discovered by the SRN apply to orthography and due to the close relation between Dutch orthography and phonology, we might compare them to the real phonotactic rules.

5 Conclusions

Looking for a connectionist architecture capable of modelling the phonotactics of natural language words, we have explored SRNs. In spite of some earlier unsuccessful experiments with SRN phonotactics modelling, the promising SRN theoretical research encouraged us to continue the search for effective modelling. The temporal information in the context layer is enough for the network to produce correct predictions. This is expected, since SRNs can simulate any frontier-to-root automata as some recent theoretical research shows.

The experiments we conducted confirmed our expectations. A sufficiently trained network predicted the possible successors after a given left context with reasonable error rate. That is, reading a word symbol by symbol, the network predicted correctly which symbols can follow the sub-word presented so far.

The success of the experiments was due to a modified interpretation of the output neurons, which was not well developed until now. In order to judge if a given character is a possible successor, we compared its activation with a threshold, which we set after training. Consequently, a word was accepted as belonging to the training language if all its letters were recognised as possible successors. Another mechanism that improved the learning was to stress more frequent combinations, i.e. more frequent words. Finally, the use of some negative data seems to have helped as well in setting the threshold optimally.

In the present study orthographic word representation was used, but similar results would be expected if phonetic representation had been studied. The difference would be the number of the basic tokens, which would be larger in the second case. In spite of this, the method should be able to study the phonetic patterns, which are simpler and better understood than those based on orthographic representation.

Since we achieved successful learning with a small SRN, we did not pay too much attention to the learning algorithm – a network of size 5 to 10 hidden neurons achieved an acceptable error rate. However, other linguistic applications may require larger networks and longer training. In that case we suggest further improvement of the learning algorithm. There are number of techniques that accelerate the learning of BP and BPTT learning algorithms. The topology might be changed dynamically as well. Evolutionary programming techniques provide interesting methods of optimal topology selecting. There are other methods that increase or decrease the hidden layer size during the training (Haykin, 1994; Stoianov, 1994). If these techniques are applied to the phonotactics problem, we think that the performance would improve further.

The successful learning of a non-trivial natural language wordlist in SRNs – the main contribution of the work presented here – evidently confirms that parallel distributed processes can attack complex problems in NLP, a claim that could be made earlier primarily for symbolic and stochastic approaches.

Acknowledgements

We thank Wouter Jansen for helpful advice on phonology. We thank also Hans van Halteren and the anonymous reviewers for valuable comments and suggestions. The work is supported by BCN grant.

References

Ainsworth-Darnell, Kim, H. Shulman and J.E. Boland. (1998). Dissociating Brain Responses to Syntactic and Semantic Anomalies: Evidence from Event-Related Potentials, in *Journal of Memory and Language*, 38, 112-130.

Cairns, P., R. Shillcock, N. Chater and J. Levy (1997). Bootstrapping Word Boundaries: A Bottom-up Corpus-Based Approach to Speech Segmentation, in *Cognitive Psychology*, 33(2), 111-153.

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

Cohen, A., C.L. Ebeling, K. Fokkema and A.G.F. van Holk (1972). *Fonologie van het Nederlands en het Fries.* Martinus Nijhoff, The Hague.

Dell, Gary S., C. Juliano and Anita Gouvindjee (1993). Structure and Content in Language Production: A Theory of Frame Constraints in Phonological Speech Errors, in *Cognitive Science*, 17, 149-195.

Dorffner, Georg (1991). "Radical" Connectionism for Natural Language Processing, in *Proceedings of the Spring Symposium on Connectionist NLP*. Standford, CA, 95-106.

Doya, K. (1993). Universality of fully connected recurrent neural networks. Technical report, University of California, San Diego.

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

Elman, Jefrey L., E. Bates, M.H. Johnson, A. Karmiloff-Smith, D. Parisi and Kim Plunket (1996). *Rethinking Innates (A connectionist perspective on development).* A Bradford Book, MIT Press.

Haykin, Simon (1994). *Neural Networks.* Macmillan College Publications.

Hirshman, Elliot and E. Jackson (1997). Distinctive Perceptual Processing and Memory, in *Journal of Memory and Language*, 36, 2-12.

Hornik, K., M. Stinchcombe and H. White (1989). Multilayer feedforward networks are universal approximators, in *Neural Networks*, 2, 359-366.

Kandel, E.C., J.H. Schwartz and T.M. Jessell (1991). *Principles of Neural Science.* Appleton & Lange.

Laver, John (1994). *Principles of phonetics.* Cambridge University Press.

Lawrence, S., C.L. Giles and S. Fong (1995). On the Applicability of Neural Networks and Machine Learning Methodologies to Natural Language

Processing. Technical Report UMIACS-TR-95-64 and CS-TR-3479, University of Maryland.

Lawrence, Steve, S. Fong and C. Lee Giles (1996). Natural Language Grammatical Inference: A Comparison of Recurrent Neural Networks and Machine Learning Methods, in *Connectionist, statistical and symbolic approaches to learning for Natural Language Processing.* Springer-Verlag, 33-47.

Miikkulainen, Risto and M. Dyer (1991). Natural Language Processing with Modular PDP Network and Distributed Lexicon, in *Cognitive Science*, 15(3), 343-399.

Nerbonne, John et al. (1996). Phonetic distance between Dutch dialects, in G. Dueux, W. Daelemans and S. Gillis (eds.). *Papers from Computational Linguistics in the Netherlands 1995*, 185-202.

Plaut, D.C., J. McClelland, M. Seidenberg and K. Patterson (1996). Understanding Normal and Impaired Word Reading: Computational Principles in Quasi-Regular Domains, in *Psychological Review*, 103, 56-115.

Reilly, Ronan G. (1995). Sandy Ideas and Coloured Days: Some Computational Implications of Embodiment, in *Artificial Intelligence Review*, 9, 305-322.

Rodd, Jenifer (1997). Recurrent Neural-Network Learning of Phonological Regularities in Turkish, in *Proceedings of the International Conference on Computational Natural Language Learning*, Madrid, July 1997, 97-106.

Rumelhart, D.E., G.E. Hinton and R.J. Williams (1986). Learning internal representations by error propagation, in David E. Rumelhart and James A. McClelland (eds.) *PDP.* Vol 1. MIT Press, Cambridge, MA.

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

Sperduti, Alissandro (1997). On the Computational Power of Recurrent Neural Networks for Structures, in *Neural Networks*, 10(3), 395-400.

Shillcock, Richard, Joe Levy, Geoff Lindsey, Paul Cairns and Nick Chater (1993). Connectionist Modelling of Phonological Space, in Mark Ellison and J. Scobbie (eds.) *Computational Phonology*, 179-195.

Stoianov, I.P, I. Baruch and E. Gortcheva (1994). Off-line Signature verification by Neural Network pressure analysis, in *Proceedings of the International Conference NNACIP'94*, 74-79.

Stoianov, Ivelin P. (1998). Tree-based Analysis of Simple Recurrent Network Learning, in *Proceedings of COLING-ACL'98.* Montreal, Canada, vol. 2, 1502-1504.

Stowe, Laurie A. et al. (1994). PET Studies of Language: An Assessment of the Reliability of the Technique, in *Journal of Psycholinguistic Research*, Plenum Publ. Corp., 23(6).

Tjong Kim Sang, Erik F. (1995). The Limitations of Modelling Finite State Grammars with Simple Recurrent Networks, in *Proceedings of the 5th CLIN meeting*, Enschede, The Netherlands, 133-143.

Tjong Kim Sang, Erik F. (1998). Machine Learning of Phonotactics. PhD Thesis. University of Groningen, Faculty of Arts.

Wan, Erik and F. Beufays (1996). Diagrammatic Derivation of Gradient Algorithms for Neural Networks, in *Neural Computations*, 8(1), 182-201.

Appendix A. Simple Recurrent Networks

In order to facilitate the reader interested in implementing a connectionist natural language processing system, we provide a technical description of SRNs and the BPTT learning algorithm.

The forward pass for the hidden layer is computed in accordance with (1,2):

(1) $net^H_i(t) = \Sigma_{j=1..|InputLayer|} w^H_{ij} in_j(t) + \Sigma_{k=1..|ContextLayer|} w^H_{ik} cn_k(t)$

(2) $hn_i(t) = f(net^H_i(t))$

where $net^H_i(t)$ is a sum of the activity provided to the inputs of the *i*th hidden neuron (i=1.. |HiddenLayer|) at time t. $in_j(t)$, $cn_k(t)$ and $hn_i(t)$ are the activation of the *j*th input, *k*th context, and *i*th hidden neurons at time t. w^H_{ij} and w^H_{ik} are the weights of the connections between *j*th input neurons, and *i*th hidden neurons, and *k*th context neurons and *i*th hidden neurons. For convenience, the bias is encoded as an extra input neuron with constant activation 1. The activation function f(.) is of sigmoidal type — logistic or hyperbolic tangent. After the activation of the hidden neurons, their activity is copied to the context neurons in accordance with (3):

(3) $cn_i(t) = (1-\beta) cn_i(t-1) + \beta hn_i(t-1)$

The transfer coefficient β is advised to be between 0.2 and 1.

The activation of the output layer depends only on the hidden layer (4,5):

(4) $net^O_i(t) = \Sigma_{j=1..|HiddenLayer|} w^O_{ij} in_j(t)$

(5) $on_i(t) = f(net^O_i(t))$

for all output neurons *i* (*i*= 1 .. |OutputLayer|).

The BPTT learning algorithm starts with computing the error and deltas at the output layer and the updates of the weights connecting the hidden layer to output layer (6,7):

(6) $\delta_i^O(t) = f'(net^O_i(t)) (C_i(t) - on_i(t))$

(7) $\Delta w^O_{ij} = \eta . \Sigma_{t=(t0+1)..(t0+|word|)} \delta_i^O(t) hn_j(t)$

where $C_i(t)$ stands for the desired activation of the output neuron *i* (*i*=1..|OutputLayer|) at time t (see section 3 for details). Index *j* stands for the hidden neurons (*j*=1..|HiddenLayer|). Provided that the activation function f(x) is the logistic function $f(x)=(1+e(-x))^{-1}$, the derivative of f(x) is $f'(x)=f(x)(1-f(x))$. The deltas and updates of the weights connecting hidden layer to input layer and the context layer are computed in accordance to (8,9):

(8) $\delta_i^H(t) = f'(net^H_i(t))[\Sigma_{j=1..|OutputLayer|} w^O_{ji}\ \delta_j^O(t) + \Sigma_{k=1..|ContextLayer|} w^H_{ik} \delta_k^H(t+1)]$

(9) $\Delta w^H_{ij} = \eta\ \Sigma_{t=(t0+1)..(t0+|word|)}\ \delta_i^H(t)\ in_j(t-1)$

where i=1..|HiddenLayer|, j=1..(|InputLayer|+|ContextLayer|) and n(t) is the joined vector containing both in(t) and cn(t). The second sum in (8) represents the context layer delta-term $\delta_k^C(t+1)$, computed by back-propagating the delta $\delta_i^H(t+1)$ through the weights connecting the context neurons to the hidden neurons.

We note that (6) and (8) have to be computed for an earlier time cycle. Also, the weight updating (7) and (9) is performed after presenting all characters from a given word and computing the deltas for each time step at which a given word was processed.

To speed-up the training, a momentum-term also can be applied with (7) and (9). This term helps the net to escape local minima, which the net can meet quite often running over the error surface. For further reading about SRN, BP and BPTT, we refer to Haykin (1994) and Wan (1996); an optimization of the learning process is given in Stoianov (1994).

Appendix B. A part of the Dutch wordlist

Word	Frequency	Word	Frequency	Word	Frequency
De	100	die	19	je	9
Het	52	niet	17	om	8
Van	49	is	17	haar	8
En	47	met	16	naar	8
Een	46	was	13	ook	7
In	35	voor	12	dan	7
Dat	26	maar	11	door	7
Te	22	als	11	of	7
Ik	20	aan	11	had	7
Hij	20	er	10		