Learning Phonotactics with Simple Processors[*]

John Nerbonne[1] and Ivilin Stoianov[2]
[1] Alfa-informatica, University of Groningen, The Netherlands
[2] Department of Psychology, University of Padova, Italy

Abstract

This paper explores the learning of phonotactics in neural networks. Experiments are conducted on the complete set of over $5,000$ Dutch monosyllables extracted from CELEX, and the results are shown to be accurate within 5% error. Extensive comparisons to human phonotactic learning conclude the paper. We focus on whether phonotactics can be effectively learned and how the learning which is induced compares to human behavior.

1 Introduction

PHONOTACTICS concerns the organization of the phonemes in words and syllables. The phonotactic rules constrain how phonemes combine in order to form larger linguistic units (syllables and words) in that language (Laver 1994). For example, Cohen, Ebeling & van Holk (1972) describes the phonemes combinations possible in Dutch, which will be the language in focus in this study.

Phonotactic rules are *implicit* in natural languages so that humans require no explicit instruction about which combinations are allowed and which are not. An explicit phonotactic grammar can of course be abstracted from the words in a language, but this is an activity linguists engage in, not language learners in general. Children normally learn a language's phonotactics in their early language development and probably update it only slightly once they have mastered the language.

Most work on language acquisition has arisen in linguistics and psychology, and that work employs mechanisms that have been developed for language,

[*]The authors are particularly pleased to offer this piece to a *Festschrift* honoring Dr. Dr. h.c. Tjeerd de Graaf, who graciously agreed to cooperate in the supervision of Stoianov's Ph.D. project 1997-2001 at the University of Groningen. Even if Tjeerd is best known for his more recent work on descriptive linguistics, minority languages and language documentation, his early training in physics and earlier research on acoustic phonetics made him one of the best-suited supervisors for projects such as the one reported on here involving advanced learning algorithms. Tjeerd's sympathy with Eastern Europe languages and cultures is visceral and might have led him to agree in any case, but we particularly appreciated his phonetic acumen.

typically, discrete, symbol-manipulation systems. Phonotactics in particular has been modeled with n-gram models, Finite State Machines, Inductive Logic Programming, etc. (Tjong Kim Sang 1998, Konstantopoulos 2003). Such approaches are effective, but a cognitive scientist may ask whether the same success could be possible using less custom-made tools. The brain, viewed as a computational machine, exploits other principles, which have been modeled in the approach known as Parallel Distributed Processing (PDP), which were thoroughly described in the seminal work of Rumelhart & McClelland (1986). Computational models inspired by the brain structure and neural processing principles are Neural Networks (NNs), also known as connectionist models.

Learning phonotactic grammars is not an easy problem, especially when one restricts one's attention to cognitively plausible models. Since languages are experienced and produced dynamically, we need to focus on the processing of sequences, which complicates the learning task. The history of research in connectionist language learning shows both successes and failures even when one concentrates on simpler structures, such as phonotactics (Stoianov, Nerbonne & Bouma 1998, Stoianov & Nerbonne 2000, Tjong Kim Sang 1995, Tjong Kim Sang & Nerbonne 1999, Pacton, Perruchet, Fayol & Cleeremans 2001).

This paper will attack phonotactics learning with models that have no specifically linguistic knowledge encoded *a priori*. The models naturally *do* have a bias, viz., toward extracting *local* conditioning factors for phonotactics, but we maintain that this is a natural bias for many sorts of sequential behavior, not only linguistic processing. A first-order Discrete Time Recurrent Neural Network (DTRNN) (Carrasco, Forcada & Neco 1999, Tsoi & Back 1997) will be used—the SIMPLE RECURRENT NETWORK (SRNs) (Elman 1988). SRNs have been applied to different language problems (Elman 1991, Christiansen & Chater 1999, Lawrence, Giles & Fong 1995), including learning phonotactics (Shillcock, Levy, Lindsey, Cairns & Chater 1993, Shillcock, Cairns, Chater & Levy 1997). With respect to phonotactics, we have also contributed reports (Stoianov et al. 1998, Stoianov & Nerbonne 2000, Stoianov 1998).

SRNs have been shown capable of representing regular languages (Omlin & Giles 1996, Carrasco et al. 1999). Kaplan & Kay (1994) demonstrated that the apparently context-sensitive rules that are standardly found in phonological descriptions can in fact be expressed within the more restrictive formalism of regular relations. We begin thus with a device which is in principle capable of representing the needed patterns.

We then simulate the language learning task by training networks to produce context-dependent predictions. We also show how the continuous predictions of trained SRNs—likelihoods that a particular token can follow the current context—can be transformed into more useful discrete predictions, or, alternatively, string recognitions.

In spite of the above claims about representability, the Back-Propagation (BP) and Back-Propagation Through Time (BPTT) learning algorithms used to train SRNs do not always find optimal solutions—SRNs that produce only correct context-dependent successors or recognize only strings from the training language. Hence, section 3 focuses on the practical demonstration that a realistic language learning task may be simulated by an SRN. We evaluate the network learning from different perspectives—grammar learning, phonotactics learning, and language recognition. The last two methods need one language-specific parameter—a *threshold* —that distinguishes successors/words allowed in the training language. This threshold is found with a post-training procedure, but it can also be sought interactively during training.

Finally, section 4 assesses the networks from linguistic and psycholinguistic perspectives: a static analysis extracts acquired linguistic knowledge from network weights, and the network performance is compared to humans' in a lexical decision task. The network performance, in particular the distribution of errors as a function of string position, will be compared to alternative construals of Dutch syllabic structure—following a suggestion from discussions of psycholinguistic experiments about English syllables (Kessler & Treiman 1997).

1.1 Motivations for a Phonotactic Device

This section will review standard arguments that demonstrate the cognitive and practical importance of phonotactics. English phonotactic rules such as:

'/s/ may precede, but not follow /t/ syllable-initially'

(ignoring loanwords such as 'tsar' and 'tse-tse') may be adduced by judging the well-formedness of sequences of letters/phonemes, taken as words in the language, e.g. /stɔp/ vs. */tsɔp/. There may also be cases judged to be of intermediate acceptability. So, even if all of the following are English words:

/mʌðəɹ/ 'mother', /fɑðəɹ/ 'father', /sɪstəɹ/ 'sister'

None of the following are, however:

*/mðəɹ/, */fɑæəɹ/, */tssɪəɹ/

None of these sound like English words. However, the following sequences:

/mɪðən/, /fuðəɹ/, /sɑntəɹ/

"sound" much more like English, even if they mean nothing and therefore are not genuine English words. We suspect that, e.g., /sɑntəɹ/ 'santer', could be used to name a new object or a concept.

This simple example shows that we have a feeling for word structure, even if no explicit knowledge. Given the huge variety of words, it is more efficient to put this knowledge into a compact form—a set of phonotactic rules. These rules would state which phonemic sequences sound correct and which do not. In this same vein, second language learners experience a period when they recognize that certain phonemic combinations (words) belong to the language they learn without knowing the meaning of these words.

Convincing psycholinguistic evidence that we make use of phonotactics comes from studying the information sources used in word segmentation

(McQueen 1998). In a variety of experiments, this author shows that word boundary locations are likely to be signaled by phonotactics. The author rules out the possibility that other sources of information, such as prosodic cues, syllabic structure and lexemes, are sufficient for segmentation. Similarly, Treiman & Zukowski (1990) had shown earlier that phonotactics play an important role in the syllabification process. According to McQueen (1998), phonotactic and metrical cues play complementary roles in the segmentation process. In accordance with this, some researchers have elaborated on a model for word segmentation: THE POSSIBLE WORD CONSTRAINTS (Norris, McQueen, Cutler & Butterfield 1997), in which likely word-boundary locations are marked by phonotactics, metrical cues, etc., and in which they are further fixed by using lexicon-specific knowledge.

Exploiting the specific phonotactics of Japanese, Dupoux, Pallier, Kakehi & Mehler (2001) conducted an experiment with Japanese listeners who heard stimuli that contained illegal consonant clusters. The listeners tended to hear an acoustically absent vowel that brought their perception into line with Japanese phonotactics. The authors were able to rule out lexical influences as a putative source for the perception of the illusory vowel, which suggests that speech perception must use phonotactic information directly.

Further justification for the postulation of a neurobiological device that encodes phonotactics comes from neurolinguistic and neuroimaging studies. It is widely accepted that the neuronal structure of the Broca area (in the brain's left frontal lobe) is used for language processing, and more specially that it represents a general sequential device (Stowe, Wijers, Willemsen, Reuland, Paans & Vaalburg 1994, Reilly 2002). A general sequential processor capable of working at the phonemic level would be a plausible realization of a neuronal phonotactic device.

Besides cognitive modeling, there are also a number of practical problems that would benefit from effective phonotactic processing. In speech recognition, for example, a number of hypotheses that explain the speech signal are created, from which the impossible sound combinations have to be filtered out before further processing. This exemplifies a LEXICAL DECISION task, in which a model is trained on a language L and then tests whether a given string belongs to L. In such a task a phonotactic device would be of use. Another important problem in speech recognition is word SEGMENTATION. Speech is continuous, but we divide it into psychologically significant units such as words and syllables. As noted above, there are a number of cues that we can use to distinguish these elements—prosodic markers, context, but also phonotactics. Similarly to the former problem, an intuitive strategy here is to split the phonetic/phonemic stream at the points of violation of phonotactic constraints (see Shillcock et al. (1997) and Cairns, Shillcock, Chater & Levy (1997) for connectionist modeling). Similarly, the constraints of the letters forming words in written languages (GRAPHOTACTICS) is useful in word processing applications, for example, SPELL-CHECKING.

There is another, more speculative aspect to investigating phonotactics. Searching for an explanation of the structure of the natural languages, Carstairs-

McCarthy presented in his recent book (1999) an analogy between syllable structure and sentence structure. He argues that sentences and syllables have a similar type of structure. Therefore, if we find a proper mechanism for learning the syllabic structures, we might apply a similar mechanism to learning syntax as well. Of course, syntax is much more complex and more challenging, but if Carstairs-McCarthy is right, the basic principles of both devices might be the same.

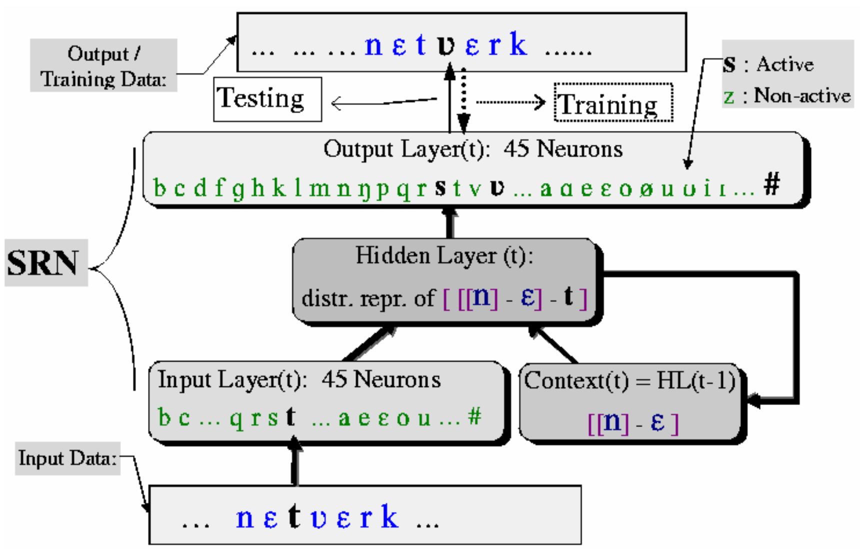
2 Simple Recurrent Networks

This section will briefly present Simple Recurrent Networks (Elman 1988, Robinson & Fallside 1988) and will review earlier studies of sequential, especially phonotactic learning. Detailed descriptions of the SRN processing mechanisms and the Back-Propagation Through Time learning algorithm that is used to train the model are available elsewhere (Stoianov 2001, Haykin 1994), and will be reviewed only superficially

Simple Recurrent Networks (SRNs) were invented to encode simple artificial grammars, as an extension of the Multilayer Perceptron (Rumelhart, Hinton & Williams 1986) with an extra input—a context layer that holds the hidden layer activations at the previous processing cycle. After training, Elman (1988) conducted investigations on how context evolves in time. The analysis showed graded encoding of the input sequence: similar items presented to the input were clustered at close, but different, shifting positions. That is, the network discovered and implicitly represented in a distributed way the rules of the grammar generating the training sequences. This is noteworthy, because the rules for context were not encoded, but rather acquired through experience. The capacity of SRNs to learn simple artificial languages was further explored in a number of studies (Cleeremans, Servan-Schreiber & McClelland 1989, Gasser 1992).

SRNs have the structure shown in Figure 1. They operate as follows: Input sequences S^I are presented to the *input* layer, one element $S^I(t)$ at a time. The purpose of the input layer is just to transfer activation to the *hidden* layer through a weight matrix. The hidden layer in turn copies its activations after every step to the *context* layer, which provides an additional input to the hidden layer—i.e., information about the past, after a brief delay. Finally, the hidden layer neurons output their signal through a second weight matrix to the *output* layer neurons. The activation of the latter is interpreted as the product of the network. Since the activation of the hidden layer depends both on its previous state (the context) and on the current input, SRNs have the theoretical capacity to be sensitive to the entire history of the input sequence. However, practical limitations restrict the time span of the context information to, maximally 10-15 steps (Christiansen & Chater 1999). The size of the layers does not restrict the range of temporal sensitivity.

The network operates in two working regimens—supervised training and network use. In the latter, the network is presented the sequential input data $S^I(t)$ and computes the output $N(t)$ using contextual information. The training



Output / Training Data:
… … … n ɛ t ʋ ɛ r k ……
Testing
Training
s : Active
z : Non-active
Output Layer(t): 45 Neurons
b c d f g h k l m n ŋ p q r s t v ʋ … a ɑ e ɛ o ø u ʊ i ɪ … #
SRN
Hidden Layer (t):
distr. repr. of [[[n] - ɛ] - t]
Input Layer(t): 45 Neurons
b c … q r s t … a e ɛ o u … #
Context(t) = HL(t-1)
[[n] - ɛ]
Input Data:
… n ɛ t ʋ ɛ r k …

Figure 1: Learning phonotactics with the SRNs. If the training data set contains the words /nɛt#/, /nɛts#/ and /nɛtʋərk#/ then after the network has processed a left context /nɛ/, the reaction to an input token /t/ will be active neurons corresponding to the symbol '#' and the phonemes /s/, and /ʋ/.

regimen involves the same sort of processing as network use and also includes a second, training step, which compares the network reaction $N(t)$ to the desired one $S^T(t)$, and which uses the difference to adjust the network behavior in a way that improves future network performance on the same data.

The two most popular supervised learning algorithms used to train SRNs are the standard Back-Propagation algorithm (Rumelhart et al. 1986) and the Back-Propagation Through Time algorithm (Haykin 1994). While the earlier is simpler because it uses information from one previous time step only (the context activation, the current network activations, and error), the latter trains the network faster, because it collects errors from all time steps during which the network processes the current sequence and therefore it adjusts the weights more precisely. However, the BPTT learning algorithm is also cognitively less plausible, since the collection of the time-spanning information requires mechanisms specific for the symbolic methods. Nevertheless, this compromise allows more extensive research, and without it the problems discussed below would require much longer training time when using standard computers for simulations. Therefore, in the experiments reported here the BPTT learning algorithm will be used. In brief, it works in the following way: the network reaction to a given input sequence is compared to the desired target sequence at every time step and an error is computed. The network activation and error at each step are kept in a stack. When the whole sequence is processed, the error is propagated back through space (the layers) and time, and weight-updating values are computed. Then, the network weights are adjusted with the values computed in this way.

2.1 Learning Phonotactics with SRNs

Dell, Juliano & Govindjee (1993) showed that words could be described not only with symbolic approaches, using word structure and content, but also by a connectionist approach. In this early study of learning word structure with neural nets (NNs), the authors trained SRNs to predict the phoneme that follows the current input phoneme, given context information. The data sets contained 100 - 500 English words. An important issue in this paper was the analysis and modeling of a number of speech-error phenomena, which were taken as strong support for parallel distributed processing (PDP) models, in particular SRNs. Some of these phenomena were: phonological movement errors (*reading list* - l*eading list*), manner errors (*department* - j*epartment*), phonotactic regularity violations (*dorm* - *dlorm*), consonant-vowel category confusions and initial consonant omissions (cluster-initial consonants drop more often than non-initial ones, so 'stop' is mispronounced [tɔp]).

Aiming at segmentation of continuous phonetic input, Shillcock et al. (1997) and Cairns et al. (1997) trained SRNs with a version of the BPTT learning algorithm on English phonotactics. They used 2 million phonological segments derived from a transcribed speech corpus and encoded with a vector containing nine phonological features. The neural network was presented a single phoneme at a time and was trained to produce the previous, the current and the next phonemes. The output corresponding to the predicted phoneme was matched against the following phoneme, measuring cross-entropy; this produced a varying error signal with occasional peaks corresponding to word boundaries. The SRN reportedly learned to reproduce the current phoneme and the previous one, but was poor at predicting the following phoneme. Correspondingly, the segmentation performance was quite modest, predicting only about one-fifth of the word boundaries correctly, but it was more successful in predicting syllable boundaries. It was significantly improved by adding other cues such as prosodic information. This means that phonotactics might be used alone for syllable detection, but polysyllabic word detection needs extra cues.

In another connectionist study on phonological regularities, Rodd (1997) trained SRNs on 602 Turkish words; the networks were trained to predict the following phonemes. Analyzing the hidden layer representations developed during the training, the author found that hidden units came to correspond to graded detectors for natural phonological classes such as vowels, consonants, voiced stops and front and back vowels. This is further evidence that NN models can capture important properties of the data they have been trained on without any prior knowledge, based only on statistical co-occurrences.

Learning the graphotactics and phonotactics of Dutch monosyllables with connectionist models was first explored by Tjong Kim Sang (1995) and Tjong Kim Sang & Nerbonne (1999), who trained SRNs to predict graphemes/phonemes based on preceding segments. The data was orthogonally encoded, that is, for each phoneme or grapheme there was exactly one neuron activated at the input and output layers (see below § 3.1). To test the knowledge learned by the network, Tjong Kim Sang and Nerbonne tested whether the activation of the neurons corresponding to the expected symbols are greater than a threshold determined as the lowest activation for some correct sequence encountered during the training data. This resulted in almost perfect acceptance of unseen Dutch words (generalization), but also in negligible discrimination with respect to (ill-formed) random strings. The authors concluded that "SRNs are unfit for processing our data set" (Tjong Kim Sang & Nerbonne 1999).

These early works on learning phonotactics with SRNs prompted the work reported here. First, Stoianov et al. (1998) demonstrated that the SRNs in Tjong Kim Sang and Nerbonne's work were learning phonotactics rather better than those authors had realized. By analyzing the error as a function of the acceptance threshold, Stoianov et al. (1998) were able to demonstrate the existence of thresholds successful at both the acceptance of well-formed data and the rejection of ill-formed data (see below § 3.6.2 for a description of how we determine such thresholds). The interval of high-performing thresholds is narrow, which is why earlier work had not identified it (see Fig. 2 on how narrow the window is). More recently, Stoianov & Nerbonne (2000) have studied the performance of SRNs from a cognitive perspective, attending to the errors produced by the network and to what extent it correlates with the performance of humans on related lexical decision tasks. The current article ties these two strands of work and presents it systematically.

3 Experiments

The challenge in connectionist modeling is not only developing theoretical frameworks, but also obtaining the most from the network models during experimentation. This section focuses on experiments on learning the phonotactics of Dutch syllables with Simple Recurrent Networks and discusses a number of related problems. It will be followed by a study on the network behavior from a linguistic point of view.

3.1 Some implementation decisions

SRNs were presented in section 2. A first implementation decision concerns how sounds are to be represented. A simple ORTHOGONAL strategy is to choose a vector of n neurons to represent n phonemes, to assign each phoneme (e.g. /ə/) to a neuron (e.g., neuron 5 in a sequence of 45), and then to activate that one neuron and deactivate all the others whenever the phoneme is to be represented (so a /ə/ is represented by four deactivated neurons, a single activated one, and then forty more deactivated neurons). This orthogonal strategy makes no assumptions about phonemes being naturally grouped into classes on the basis of linguistic FEATURES such as consonant/vowel status, voicing, place of articulation, etc. An alternative strategy exploits such features by assigning each feature to a neuron and then representing a phoneme via a translation of its feature description into a sequence of corresponding neural activations.

In phonotactics learning, the input encoding method might be feature-based or orthogonal, but the output decoding should be orthogonal in order to obtain a simple prediction of successors, and to avoid a bias induced from the peculiarities of the feature encoding scheme used. The input encoding chosen was also orthogonal, which also requires the network discover natural classes of phonemes by itself.

The orthogonal encoding implies that we need as many neurons as we have phonemes, plus one for the end-of-word '#' symbol. That is, the input and output layers will have 45 neurons. However, it is usually difficult to choose the right size of the hidden layer for a particular learning problem. That size is rather indirectly related to the learning task and encoding chosen (as a subcomponent of the learning task). A linguistic bias in the encoding scheme, e.g., feature-based encoding, would simplify the learning task and decrease the number of the hidden neurons required to learn it (Stoianov 2001). Intuition tells us that hidden layers that are too small lead to an overly crude representation of the problem and larger error. Larger hidden layers, on the other hand, increase the chance that the network wanders aimlessly because the space of possibilities it needs to traverse is too large. Therefore, we sought an effective size in a pragmatic fashion. Starting with a plausible size, we compared its performance to nets with double and half the number of neurons in the hidden layer. We repeated in the direction of the better behavior, keeping track of earlier bounds in order to home in on an appropriate size. In this way we settled on a range of 20–80 neurons in the hidden layer, and we continued experimentation on phonotactic learning using only nets of this size.

However, even given the right size of the hidden layer, the training will not always result in an optimal weight set W^* since the network learning is nondeterministic—each network training process depends on a number of stochastic variables, e.g., initial network weights and an order of presentation of examples. Therefore, in order to produce more successful learning, several SRNs with different initial weights will be trained in a pool (group).

The back-propagation learning algorithm is controlled by two main parameters—a learning coefficient η and a smoothing parameter α. The first one controls the speed of the learning and is usually set within the range $(0.1 \ldots 0.3)$. It is advisable to choose a smaller value when the hidden layer is larger. Also, this parameter may vary in time, starting with a larger initial value that decreases progressively in time (as suggested in Kuan, Hornik & White (1994) for the learning algorithm to improve its chances at attaining a global minimum in error). Intuitively, such a schedule helps the network approximately to locate the region with the global minima and later to make more precise steps in searching for that minimum (Haykin 1994, Reed & Marks II 1999). The smoothing parameter α will be set to 0.7, which also allows the network to escape from local minima during the search walk over the error surface.

The training process also depends on the initial values of the weights. They are set to random values drawn from a region $(-r \ldots +r)$. It is also important to find a proper value for r, since large initial weight values will produce chaotic network behavior, impeding the training. We used $r = 0.1$.

The SRNs used for this problem are schematically represented in Fig. 1, where the SRN reaction to an input sequence /nɛ/ after training on an exemplary set containing the sequences /nɛt#/, /nɛts#/, /nɛtʊərk#/ is given. For this particular database, the network has experienced the tokens '#', /s/ and /ʊ/ as possible successors to /nɛe/ during training and therefore it will activate them in response to this input sequence.

3.2 Linguistic Data - Dutch syllables

A data base L_M of all Dutch monosyllables—$5,580$ words—was extracted from the CELEX (1993) lexical database. CELEX is a difficult data source because it contains many rare and foreign words among its approximately 350,000 Dutch lexical entries, which additionally complicate the learning task. Filtering out non-typical words is a formidable task and one which might introduce experimenter prejudice, and therefore all monosyllables were used. The monosyllables have a mean length of $4.1(\sigma = 0.94, min = 2, max = 8)$ tokens and are built from a set of 44 phonemes plus one extra symbol representing space (#) used as a filler specifying *end-of-word.*

The main dataset is split into a training (L^1) and testing (L^2) databases in proportion approximately 85% to 15%. The training database will be used to train a Simple Recurrent Network and the testing one will be used for evaluating the success of word recognition. Negative data also will be created for test purposes. The complete database L_M will be used for some parts of evaluation.

In language modeling it is important to explore the frequencies of word occurrences which naturally bias humans' linguistic performance. If a model is trained on data in proportion to its empirical frequency, this focuses the learning on the more frequent words and thus improves the performance of the model. This also makes feasible a comparison of the model's performance with that of humans performing various linguistic tasks, such as a lexical decision task. For these reasons, we used the word frequencies given in the CELEX database. Because the frequencies vary greatly ($[0 \ldots 100,000]$), we presented training data items in proportion with the natural logarithms of their frequencies, in accordance with standard practice (Plaut, McClelland, Seidenberg & Patterson 1996). This approach resulted in frequencies in a range of $[1 \ldots 12]$.

3.3 Difficulty

One way to characterize the complexity of the training set is to compute the ENTROPY of the distribution of successors, for every available left context. The entropy of a language L viewed as a stochastic process measures the average surprise value associated with each element (Mitchell 1997). In our case, the language is a set of words and the elements are phonemes, hence the appropriate entropy measures the average surprise value for phonemes c preceded by a context s. Entropy is measured for a given distribution, which in our case is the set of all possible successors. We compute entropy Entr(s) for a given context s with (1):

$$\text{Entr}(s) = -\sum_{c \in \alpha} p_s(c) log_2(p_s(c)) \tag{1}$$

where α is the alphabet of segment symbols, and $p(c)$ the probability of a given context. Then the average entropy for all available contexts $s \in L$, weighted with their frequencies, will be the measure of the complexity of the words. The smaller this measure, the less difficult are the words. The maximal possible value for one context would be $log_2(45)$, that is, 5.49, and this would only obtain for the unlikely case that each phoneme was equally likely in that context. The actual average value of the entropy measured for the Dutch monosyllables, is 2.24, σ =1.32. The minimal value was 0.0, and the maximal value was 3.96. These values may be interpreted as follows: The minimal value of 0.0 means that there are left contexts with only one possible successor ($log_2(1) = 0$). A maximal value of 3.96 means that there is a one context which is as unpredictable as one in which $2^{3.96} = 16$ successors were equally likely. The mean entropy is 2.24, which is to say that in average 4.7 phonemes follow a given left context.

3.4 Negative Data

We noted above that negative data is also necessary for evaluation. Since we are interested in models that discriminate more precisely the strings from L (the Dutch syllables), the negative data for the following experiments will be biased toward L.

Three negative testing sets were generated and used: First, a set R_M containing strings with syllabic form $[C]^{0...4}V[C]^{0...3}$, based on the empirical observation that the Dutch syllables have up to four onset (initial) consonants and up to three coda (final) consonants. The second group consists of three sub-sets of R_M: $\{R_M^1, R_M^2, R_M^{3+}\}$, with fixed distances of the random strings to any existing Dutch word at 1, 2, and 3+ phonemes, respectively (measured by edit distance (Nerbonne, Heeringa & Kleiweg 1999)). Controlling for the distance to any training word allows us to assess more precisely the performance of the model. And finally, a third group: random strings built of concatenations of n-grams picked randomly from Dutch monosyllables. In particular, two sets—R_N^2 and R_N^3—were randomly developed, based on bigrams and trigrams, correspondingly.

The latter groups are the most "difficult" ones, and especially R_N^3, because it consists of strings that are closest Dutch. They are also useful for the comparison of SRN methods to n-gram modeling. The corresponding n-gram models will always wrongly recognise these random strings as words from the language. Where the connectionist predictor recognises them as non-words, it outperforms the correspondent n-gram models, which are considered as benchmark models for prediction tasks such as phonotactics learning.

3.5 Training

This section reports on network training. We will add a few more details about the training procedure, then we will present pilot experiments aimed at determining the hidden layer size. The later parts will analyse the network performance.

3.5.1 Procedure

The networks were trained in a pool on the same problem, and independently of each other, with the BPTT learning algorithm. The training of each individual network was organised in epochs, in the course of which the whole training data set is presented in accordance with the word frequencies. The total of the logarithm of the frequencies in the training data base L^1_M is about $11,000$, which is also the number of presentations of sequences per epoch, drawn in a random order. Next, for each word, the correspondent sequence of phonemes is presented to the input, one at a time, followed by the *end-of-sequence* marker '#'. Each time step is completed by copying the *hidden* layer activations to the *context* layer, which is used in the following step.

The parameters of the learning algorithm were as follows: the learning coefficient η started at 0.3 and dropped by 30% each epoch, finishing at 0.001; the momentum (smoothing) term $\alpha = 0.7$. The networks required 30 epochs to complete training. After this point, very little improvement is noted.

3.5.2 Pilot experiments

Pilot experiments aiming at searching for the most appropriate hidden layer size were done with 20, 40 and 80 hidden neurons. In order to avoid other non-determinism which comes from the random selection of negative data, during the pilot experiments the network was tested solely on its ability to distinguish admissible from inadmissible successors. Those experiments were done with a small pool of three networks, each of them trained for 30 epochs, which resulted in approximately 330,000 word presentations or 1,300,000 segments. The total number of individual word presentations ranged from 30 to 300, according to the individual word frequencies. The results of the training are given in Table 1, under the group of columns "Optimal phonotactics". In the course of the training, the networks typically started with a sharp error drop to about 13%, which soon turned into a very slow decrease (see Table 2, left 3 columns).

The training of the three pools with hidden layer size 20, 40 and 80, resulted in networks with similar performance, with the largest network performing best. Additional experiments with SRNs with 100 hidden neurons resulted in larger error than a network with 80 hidden neurons, so that we settled experimentally on 80 hidden neurons as the likely optimal size. It is clear that this procedure is rough, and that one needs to be on guard against premature concentration on one size model.

	Optimal Phonotactics			$\|\|SRN^L, T^L\|\|_{L_2}$		
Hidd Layer Size	SRN_1	SRN_2	SRN_3	SRN_1	SRN_2	SRN_3
20	10.57%	10.65%	10.57%	0.0643	0.0642	0.0642
40	10.44%	10.51%	10.44%	0.0637	0.0637	0.0637
80	10.00%	9.97%	10.02%	0.0634	0.0634	0.0632

Table 1: Results of a pilot study on phonotactics learning by SRNs with 20, 40, and 80 (rows) hidden neurons. Each network is independently trained on language L_M three times (columns). The performance is measured (a, left 3 columns) using the error in predicting the next phoneme, and (b, right 3 columns) using L_2 (semi-Euclidean) distance between the empirical context-dependent predictions and the network predictions for each context in the tree. Those two methods do not depend on randomly chosen negative data.

Epoch	1	2-4	5-10	11-15	16-30
Error (%)	15.0	12.0	10.8	10.7	10.5

Table 2: A typical shape of the SRN error during training. The error drops sharply in the beginning and then slowly decreases to convergence.

3.6 Evaluation

The performance of a neural predictor trained on phonotactics may be evaluated with different methods, depending on the particular task the network is applied to. In this section we evaluate the neural networks performing best during the pilot studies.

3.6.1 Likelihoods

The direct outcome of training the sequential prediction task is learning the successors' distribution. This will therefore be used as a basic evaluation method: the empirical context-dependent successor distribution $P_s^L(C)$ will be matched against the network context dependent predictions $NP_s^L(C)$. For this purpose, the output of the network will be normalized and matched against the distribution in the language data.

This procedure resulted in a mean L_2 (semi-Euclidean) distance of $0.063 - 0.064$, where the optimal value would be zero (see Table 1, right 3 columns).[1] These values are close to optimal but baseline models (completely random networks) also result in approximately 0.085 L_2 distance.

[1]The distance is related to Euclidean, but more exactly the distance between the two n-dimensional vectors is $L_2(\vec{x}, \vec{y}) = \sqrt{1/n \sum_{i=1}^{n} (x_i - y_i)^2}$.

3.6.2 Phonotactic Constraints

To evaluate the network's success in becoming sensitive to phonotactic constraints, we need first to judge how well it predicts individual phonemes. For this purpose we seek a THRESHOLD above which phonemes are predicted to be admissible and below which they are predicted to be inadmissable. This is done empirically—we perform a binary search for an optimal threshold, i.e. the threshold θ^* that minimizes the network error $E(\theta)$. The classification obtained in this fashion constitutes the network's predictions about phonotactics.

We now turn to evaluating the network's predictions: the method to evaluate the network from this point of view compares the context-dependent network predictions with the corresponding empirical distributions. For this purpose, the method described by Stoianov (2001) will be used. The algorithm traverses a TRIE (Aho, Hopcroft & Ullman 1983, 163–169), which is a tree representing the vocabulary where initial segments are the first branches. Words are paths through this data structure. The algorithm computes the performance at the optimal threshold determined using the procedure described in the last paragraph, i.e., at the threshold which determines which phonemes are admissible and which inadmissible (see also § 2.1). This approach compares the actual distribution with the learned distribution, and we normally use the complete database L_M for training and testing.

Figure 2 shows the error of SRN_1^{80} at different values of the threshold. The optimal threshold searching procedure resulted in 6.0% erroneous phoneme prediction at a threshold of 0.0175. This means that if we want to predict phonemes with this SRN, they would be accepted as allowed successors if the activation of the correspondent neurons are higher than 0.0175.

3.6.3 Word Recognition

Using an SRN trained on phoneme prediction as a *word* recognizing device shifts the focus from phoneme prediction to sequence classification. We wish to see whether it can classify sequences of phonemes into well-formed words on the one hand and ill-formed non-words on the other. To do this we need to translate the phoneme (prediction) values into sequence values. We do this by taking the sum of the phoneme error values for the sequence of phonemes in the string, normalized to correct for length effects. But to translate this sum into a classification, we again need to determine an acceptability threshold, and we use a variant of the same empirical optimization described above. The threshold arrived at for this purpose is slightly lower than the optimal threshold from the previous algorithm. This means that the network accepts more phonemes, which, however, is compensated for by the fact that a string is accepted only if all its phonemes are predicted. In string recognition it is better to increase the phoneme acceptance rate, because the chance to detect a non-word is larger when more tokens are tested.

Since the performance measure here is the mean percentage of correctly recognized monosyllables and correctly rejected random strings, we incorporate

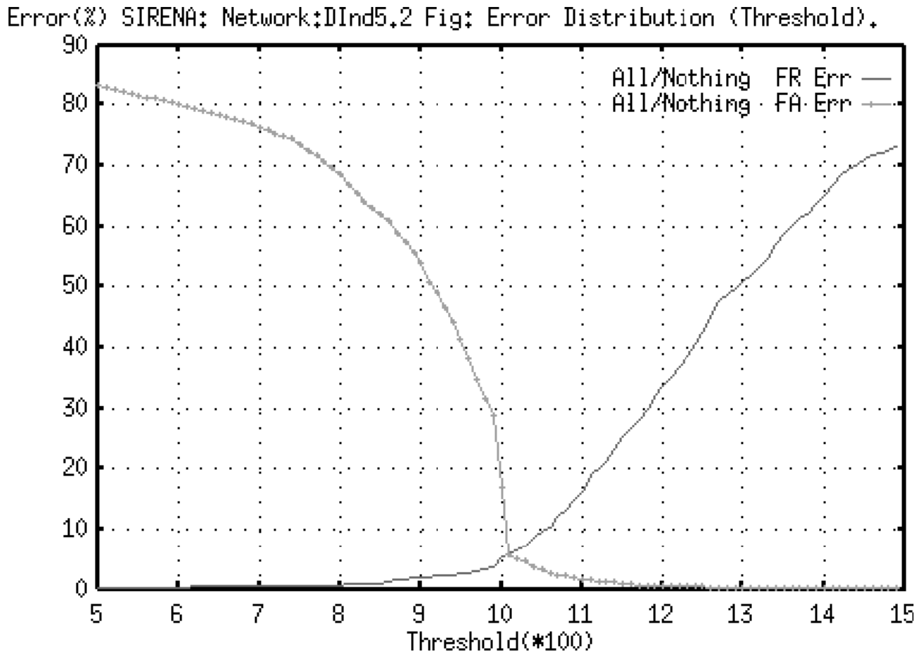


Figure 2: SRN error (in %) as a function of the threshold θ. The False Negative Error increases as the threshold increases because more and more admissible phonemes are incorrectly rejected. At the same time, the False Positive Error decreases because fewer unwanted successors are falsely accepted. The mean of those two errors is the network error, which finds its minimum 6.0% at threshold $\theta^* = 0.0175$. Notice that the optimal threshold is limited to a small range. This illustrates how critical the exact setting of threshold is for good performance.

both in seeking the optimal threshold. The negative data is as described above in § 3.4. Concerning the positive data, this approach allows us to test the generalisation capacity of the model, so that the training L_M^1 and testing L_M^2 subsets may be used here—the first for training the model and evaluating it during training, and the second to test the generalisation capacity of the trained network.

Once we determine the optimal sequence-acceptance threshold (0.016), we obtain 5% error on the positive training dataset L_M^1 and the negative strings from R_M, where the error varied $\pm 0.5\%$ depending on the random data set generated.

The model was tested further on the second group of negative data sets. As expected, strings which are more unlike Dutch resulted in smaller error. Performance on random strings from R_N^{3+} is almost perfect. In the opposite case, the strings close to real words (from R_N^1) resulted in larger error.

The generalization capabilities of the network were tested on the L_M^2 positive data, unseen during training. The error on this test set was about 6%. An explanation of the increase of the error will be presented later, when the error will be studied by varying its properties.

Another interesting issue is how SRN performance compares to other known models, e.g. n-grams. The trained SRN definitely outperformed bigrams and trigrams, which was shown by testing the trained SRNs on the non-words from R^2_N and R^3_N sets, yielding 19% and 35% error, respectively. This means that the SRN correctly rejected four out of five non-word strings composed of correct bigrams and two out of three non-word strings made of trigrams. To clarify, note that bigram models would have 100% error on R^2_N, and trigram models 100% error on R^3_N.

4 Network Analysis

The distributed representations in Neural Networks prevent the analysis of generalizations in trained models by simple observation, which symbolic learning methods allow. Smaller NNs may be analyzed to some extent by examination, but for larger networks this is practically impossible.

It is possible, however, to analyze trained networks to extract abstract knowledge about their behavior. Elman (1988), for example, trained an SRN to learn sentences and then analyzed the hidden layer activations of that SRN in various contexts, from which he showed that the network had internally developed syntactical categories. Similarly, we trained SRNs on phonotactics (Stoianov et al. 1998), and then analyzed the network statically, by viewing the weight vectors of each neuron as pattern classifiers. We showed that the SRN had induced generalizations about phonetic categories. We follow that earlier work in order to study network behavior, and we present the results of this study in the first subsection.

Another approach to the analysis of connectionist models assumes that they are black boxes and examines the variation of network performance while varying some properties of the data (Plaut et al. 1996, Stoianov, Stowe & Nerbonne 1999). For example, one can vary word frequency, length, etc., and study the network error. When modeling human cognitive functions with this approach one can compare the behavior of the cognitive system and its artificial models. For example, in phonotactics modeling, one can compare results from psycholinguistic studies of a *lexical decision task* with the network reaction. This will be subject of study in the rest of the section.

4.1 Weight Analysis

The neurons of a neural network act as pattern classifiers. The inputs selectively activate one or another neuron, depending on the weight vectors. This means that information about network structure may be extracted from the weight vectors.

In this section we will present a cluster analysis of the neurons in the output layer. For that purpose, the weight vectors of the output layer of one of the networks—SRN_2^{40} (from table 1)—were clustered by using Pearson correlation coefficients. Then, a dendrogram of the vectors was created by using a minimum variance (Ward's) method, and each vector was labeled with the phoneme it corresponds to[2]. The resulting diagram is shown in Figure 3.

We can see that the weight vectors (and correspondingly, the phonemes) cluster into known major natural classes—vowels (in the bottom) and consonants (the upper part). The vowels are split into two major categories: low and semi-low, front vowels (/ɑ, ɛ, a, e/), and high, back ones. The latter, in turn, are clustered into *round*+ and *round*− classes. Consonants appear to be categorized in a way less congruent with phonetics. But here, too, some established groups are distinguished. The first subgroup contains non-coronal consonants (/f, k, m, p, x/) with the exceptions of /l/ and /n/. Another subgroup contains voiced obstruents (/ɣ, d, ʝ, d͡ʝ/). The delimiter '#' is also clustered as a consonant, in a group with /t/, which is also natural. The upper part of the figure seems to contain phonemes from different groups, but we can recognize that most of these phonemes are quite rare in Dutch monosyllables, e.g., /ə/, perhaps because they have been 'loaned' from other languages, e.g. /g/.

[2]The cluster analysis in Figure 3 was produced by programs written by Peter Kleiweg, available at http://www.let.rug.nl/alfa

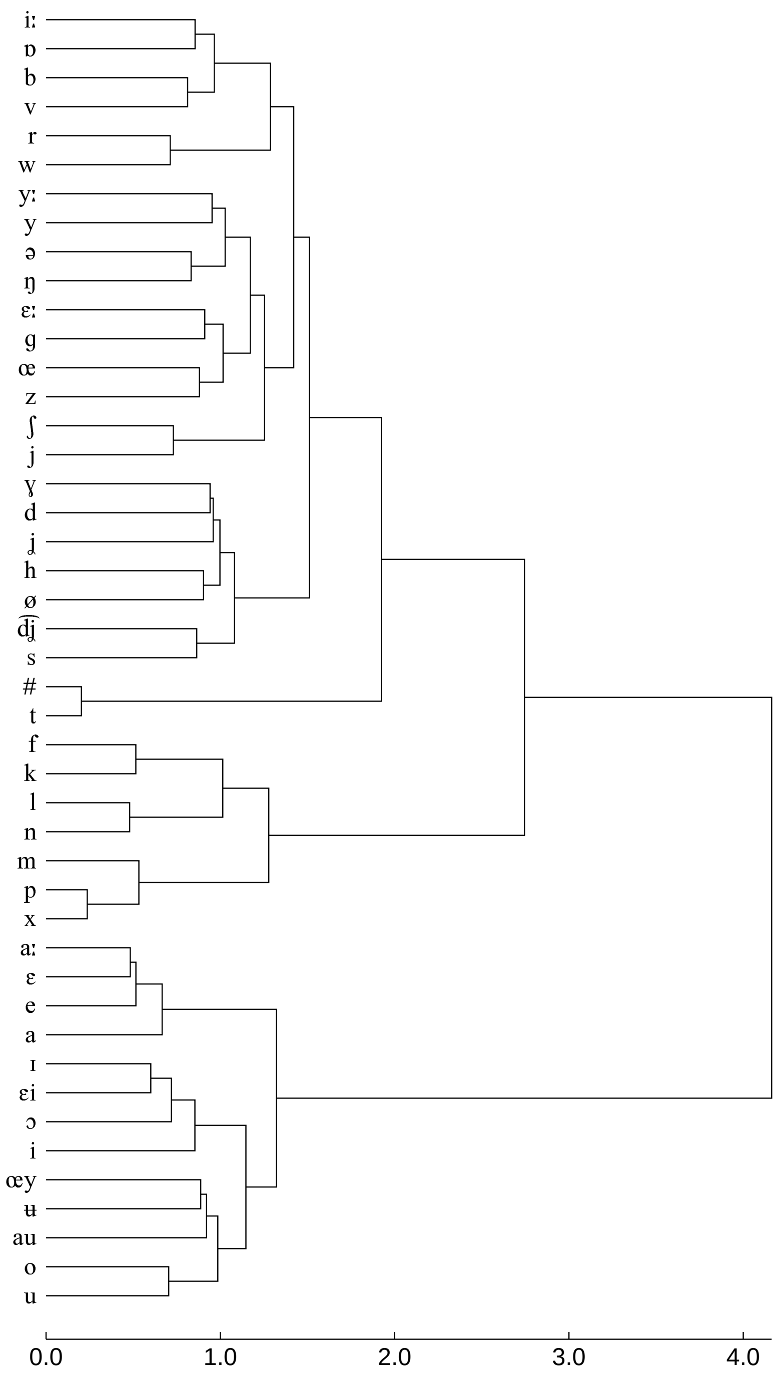


Figure 3: Cluster analysis of the vector of the output neurons, labeled with the phonemes they correspond to. The weight vectors are split into clusters which roughly correspond to existing phonetic categories.

4.2 Functional analysis

We may also study NNs by examining their performance as a function of factors such as word frequency, similarity neighborhood, and word length. Such an analysis relates computational language modeling to psycholinguistics, and we submit that it is useful to compare the models' performance with humans'. In this section we introduce several factors which have played a role in psycholinguistic theorizing. We then examine the performance of our model as a function of these factors.

4.2.1 Psycholinguistic Factors

FREQUENCY is one of the most thoroughly investigated characteristics of words that affect performance. Numerous previous studies have demonstrated that the ease and the time with which spoken words are recognized are monotonically related to the experienced frequency of words in the language environment (Luce, Pisoni & Goldinger 1990, Plaut et al. 1996). The general tendencies found are that the more frequent words are, the faster and the more precise they are recognized.

Our perception of a word is likewise known to depend on its similarity to other words. The SIMILARITY NEIGHBORHOOD of a word is defined as the collection of words that are phonetically similar to it. Some neighborhoods are dense with many phonetically similar words while others are sparse with few.

The so-called *Colthearth-N* measure of a word w counts the number of words that might be produced by replacing a single letter of w with some other. We modify this concept slightly to make it sensitive to similarity of sub-syllabic elements, so that we regard words as similar when they share two of the sub-syllables—onset, nucleus and coda. Empty onsets or codas are counted as same. The word neighborhood is computed by counting the number of the similar words. If implemented precisely, the complexity of the measuring process construed as just explained is high, so we reduce it by probing for sub-syllables rather than for units of variable size, starting from a single phoneme. This simplifies and speeds up processing. The neighborhood size of the corpus we used ranged from 0 to 77 and had mean value of $\mu = 30, \sigma = 13$.

For example, the phonological neighborhood of the Dutch word *broeds* /bruts/ is given below. Note that the neighborhood contains only Dutch words.

/brɪts/, /brots/, /bruj/, /bruit/, /bruk/, /brur/, /brus/, /brut/, /buts/,
/kuts/, /puts/, /tuts/

These represent the pronunciations of *Brits* 'British', *broods* 'bread' (gen.sg.), *broei* 'brew', *broeit* 'brew' (3rd. sg.), *broek* 'pants', *broer* 'brother', *broes* 'spray nozzle', *broed* 'brood', *boots* 'boots' (Eng. loan), *koets* 'coach', *poets* 'clean' and *toets* 'test'. Among the words with very poor neighborhood are /ʃvuŋ/ *schwung*, /bɔrts/ *boards*, /d͡ɟojnt/ *joint*, and /skʋers/ *squares*, all of which are of foreign origin. Words such as /hɛk/ *hek*, /bɑs/ *bas*, /lɑxt/ *lacht*, and /bɑkt/ *bakt* have large neighborhoods.

It is still controversial how similarity neighborhood influences cognitive processes (Balota, Paul & Spieler 1999). Intuitively, it seems likely that words with larger neighborhoods are easier to access due to many similar items, but from another perspective these words might be more difficult to access due to the nearby competitors and longer selection process. However, in the more specific *lexical decision* task, the overall activity of many candidates has been shown to facilitate lexical decisions, so we will look for the same effect here.

The property WORD LENGTH might affect performance in the lexical decision task in two different ways. On one hand, longer words provide more evidence since more phonemes are available to decide whether the input sequence is a word so that we expect higher precision for longer words, and lower precision for particularly short words. On the other hand, network error accumulating in iteration increases the error in phoneme predictions at later positions, which in turn will increase the overall error for longer words. For these reasons we expect a U-shaped patterns of dependence as word length increases. Such a pattern was observed in a study on modeling grapheme-to-phoneme conversion with SRNs (Stoianov et al. 1999). Static NNs are less likely (than dynamic models such as SRNs) to produce such patterns.

So far we have presented three main characteristics of the individual words, which we expect to affect the performance of the model. However, a statistical correlation analysis (bivariate Spearman test) showed that they are not independent, which means that an analysis of the influence of any single factor should control for the rest. In particular, there is high negative correlation between word neighborhood and word length (-0.476) , smaller positive correlation between neighborhood and frequency (0.223), and very small negative correlation between frequency and word length (-0.107). Because of the larger amount of data all these coefficients are significant at the 0.001 level.

Finally, it will be useful to seek a correlate in the simulation for REACTION TIME, which psycholinguists are particularly fond of using as a probe to understanding linguistic structure. Perhaps we can find an SRN correlate to Reaction Time (RT) for the lexical decision task in NETWORK CONFIDENCE, i.e., the amount of evidence that the test string is a word from the training language. The less confident the network, the slower the reaction, which can be implemented with a lateral inhibition (Haykin 1994, Plaut et al. 1996). The network confidence for a given word might be expressed as the product of the activations of the neurons corresponding to the phonemes of that word. A similar measure, which we call *uncertainty* U is the negative sum of (output) neuron activation logarithms, normalized with respect to word length $|w|$ (2). Note that U varies inversely with confidence. Less certain sequences get higher (positive) scores.

$$U = -\frac{1}{|w|} \sum_{i=1}^{|w|} ln(n_{c_i}) \tag{2}$$

To analyze the influence of these parameters, the network scores and U-

(a)

Frequency	Low	Mid	High
U	2.30	2.20	2.18
Error (%)	8.6	4.1	1.5

(b)

Neighb. size	Low	Mid	High
U	2.62	2.30	2.21
Error (%)	12.7	3.9	0.8

(c)

Length	Short	Mid	Long
U	2.63	2.20	2.13
Error (%)	5.2	4.4	13.1

Table 3: Effect of (a) frequency, (b) neighborhood density and (c) length effect on word uncertainty U and word error.

values were recorded for each monosyllabic word at the optimal threshold $\theta^* = 0.016$. The data was then submitted to the statistical package SPSS for analysis of variance using SPSS's General Linear Model (GLM). When analyzing network score, the analysis revealed main effects of all three parameters discussed above: word neighborhood size ($F = 18.4, p < 0.0001$), word frequency ($F = 19.2, p < 0.0001$), word length ($F = 11.5, p < 0.0001$). There was also interaction between neighborhood size and the other parameters: the interaction with word frequency had an F-score 6.6 and the interaction of the neighborhood with word length had and F-score of 4.9, both significant at 0.0001 level. Table 3 summarizes the findings. Error decreases both as neighborhood size and as frequency increases, and error dependent on length shows the predicted U-shaped form (Table 3c).

Analysis of variance on the U-values revealed similar dependencies. There were main effects of word neighborhood size ($F = 58.2, p < 0.0001$), word frequency ($F = 45.9, p < 0.0001$), word length ($F = 137.5, p < 0.0001$), as well as the earlier observed interactions between neighborhood density and the other two variables: word length ($F = 10.4, p < 0.001$) and frequency ($F = 5.235, p < 0.005$).

The frequency pattern of error and uncertainty variance was expected, given the increased evidence to the network for more frequent words. The displayed length effect showed that the influence of error gained in recursion is weaker than the effect of stronger evidence for longer words. Also, the pattern of performance when varying neighborhood density confirmed the hypothesis of the lexical decision literature that larger neighborhoods makes it easier for words to be recognized as such.

4.3 Syllabic structure

Phonotactic constraints might hint at how the stream of phonemes is organized in the language processing system. The popular phoneme, syllable and word

(a)

Word Position	2	3	4	5	6	7	8
Error(%)	4.3	1.7	1.4	0.6	0.3	0.3	0.00

(b)

Sub-syllables	Onset		Nucleus	Coda			
Relative Position	2	3	1	1	2	3	4
Error(%)	2.6	0.0	4.5	1.0	1.5	2.0	2.6

Table 4: Distribution of phoneme prediction error at a threshold of 0.016 by (a) phoneme position within words and (b) phoneme position within sub-syllables. Word and Onset positions start from 2, because the prediction starts after the first phoneme.

entities may not be the only units that we use for lexical access and production. There are suggestions that in addition, some sub-syllabic elements are involved in those processes, that is, the syllables might have not linear structure, but more complex representations (Kessler & Treiman 1997). For that purpose, we will analyze how the phoneme prediction error at a threshold of 0.016—where the network resulted in best word recognition—is located within words with respect to the following sub-syllabic elements—*onset, nucleus* and *coda.* The particular hypothesis that will be tested is whether Dutch monosyllables follow structure below that was found in English as well (Kessler & Treiman 1997).

(Onset - Rhyme (Nucleus - Coda))

The distribution of phoneme error within words (Table 4a) shows that the network makes more mistakes at the beginning than at the end of words, where SRN becomes more confident in its decision. This could be explained with increasing contextual information that more severely restricts possible phonemic combinations. A more precise analysis of the error position in the onset, the nucleus and the coda further reveals other interesting phenomena (Table 4b).

First, error within the coda increases at the coda's end. We attribute this to error accumulated toward the end of the words, as was predicted earlier. The mean entropy in the coda ($1.32, \sigma = 0.87$) is smaller than the mean entropy in the onset ($1.53, \sigma = 0.78$), where we do not observe such effects. So looser constraints are not the reason for the relatively greater error in the coda.

Next, the error at the transition onset-nucleus is much higher than the error at the surrounding positions, which means that the break between onset and rhyme (the conjunction nucleus-coda) is significant. This distribution is also consistent with the statistical finding that the entropy is larger in the *body* (the transition point onset-nucleus) ($3.45, \sigma = 0.39$), than in the rhyme ($1.94, \sigma = 1.21$). All this data support the hypothesis that onset and rhyme play significant roles in lexical access and that the syllabic structure confirmed for English by Kessler & Treiman (1997) is valid for Dutch, too.

5 Conclusions

Phonotactic constraints restrict the way phonemes combine in order to form words. These constraints are empirical and can be abstracted from the lexicon—either by extracting rules directly, or via models of that lexicon. Existing language models are usually based on abstract symbolic methods, which provide good tools for studying such knowledge. But linguistic research from a connectionist perspective can provide a fresh perspective about language because the brain and artificial neural networks share principles of computations and data representations.

Connectionist language modeling, however, is a challenging task. Neural networks use distributed processing and continuous computations, while languages have a discrete, symbolic nature. This means that some special tools are necessary if one is to model symbolic problems with connectionist models. The research reported in this paper attempted to provide answers to two basic questions: First, whether phonotactic learning is possible at all in connectionist systems, which had been doubted earlier (Tjong Kim Sang 1995, Tjong Kim Sang 1998). In the case of a positive answer, the second question is how NN performance compares to human ability. In order to draw this comparison, we needed to extract the phonotactic knowledge from a network which has learned the sequential structure. We proposed several ways of doing this.

Section 3 studied the first question. Even if there are theoretical results demonstrating that NNs have the needed finite-state capacity for phonotactic processing, there are practical limitations, so that we needed experimental support to demonstrate the practical capability of SRNs to learn phonotactics. A key to solving the problems of earlier investigators was to focus on finding a threshold that optimally discriminated the continuous neuron activations with respect to phoneme acceptance and rejection simultaneously. The threshold range at which the network achieves good discrimination is very small (see Figure 2), which illustrates how critical the exact setting of the threshold is. We also suggested that this threshold might be computed interactively, after processing each symbol, which is cognitively plausible, but we postpone a demonstration of this to another paper.

The network performance on word recognition—word acceptance rate of 95% and random string rejection rate of 95% at a threshold of 0.016—competes with the scores of symbolic techniques such as Inductive Logic Programming and Hidden Markov Models (Tjong Kim Sang 1998), both of which reflect low-level human processing architecture with less fidelity.

Section 4 addressed the second question of how other linguistic knowledge encoded into the networks can be extracted. Two approaches were used. Section 4.1 clustered the weights of the network, revealing that the network has independently become sensitive to established phonetic categories.

We went on to analyze how various factors which have been shown to play a role in human performance find their counterparts in the network's performance. Psycholinguistics has shown, for example, the ease and the time with which spoken words are recognized are monotonically related to the frequency of words in the language environment (Luce et al. 1990). The model likewise reflected the importance of neighborhood density in facilitating word recognition, which we speculated stems from the supportive evidence which more similar patterns lend to the words in their neighborhood. Whenever network and human subjects exhibit a similar sensitivity to well-established parameters, we see a confirmation of the plausibility of the architecture chosen.

Finally, the distribution of the errors within the words showed another linguistically interesting result. In particular, the network tended to err more often at the transition onset-nucleus—which is also typical for transitions between adjacent words in the speech stream and used for speech segmentation. Analogically, we can conclude from this that the conjunction nucleus-coda—rhyme—is a significant linguistic unit for the Dutch language, a result suggested earlier for English (Kessler & Treiman 1997).

We wind up this conclusion with one disclaimer and a repetition of the central claim. We have *not* claimed that SRNs are the only (connectionist) model capable of dynamic processing, nor that they are biologically the most plausible neural network. Our central claim is to have demonstrated that relatively simple connectionist mechanisms have the capacity to model and learn phonotactic structure.

References

Aho, Alfred, John Hopcroft & Jeffrey Ullman (1983), *Data Structures and Algorithms*, Addison Wesley.

Balota, David, Stephen Paul & Daniel Spieler (1999), Attentional control of lexical processing pathways during word recognition and reading, *in* S. Garrod & M. Pickering, eds, 'Studies in cognition: Language processing', UCL Press, London,England, pp. 15–57.

Cairns, Paul, R. Shillcock, Nick Chater & Joe Levy (1997), 'Bootstrapping word boundaries: A bottom-up corpus-based approach to speech segmentation', *Cognitive Psychology* **33(2)**, 111–153.

Carrasco, Rafael, Mikel Forcada & Ramon Neco (1999), 'Stable encoding of finite-state machines in discrete-time recurrent neural networks with sigmoid units', *Neural Computations* **12(9)**, 2129–2174.

Carstairs-McCarthy, Andrew (1999), *The Origins of Complex Language*, Oxford Univ Press.

CELEX (1993), 'The celex lexical data base (cd-rom)', Linguistic Data Consortium. **URL** = `http://www.kun.nl/celex`.

Christiansen, Morton H. & Nick Chater (1999), 'Toward a connectionist model of recursion in human linguistic performance', *Cognitive Science* **23**, 157–205.

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

Cohen, A., C. Ebeling & A.G.F. van Holk (1972), *Fonologie van het Nederlands en her Fries*, Martinus Nijhoff, The Hague.

Dell, Gary, Cornell Juliano & Anita Govindjee (1993), 'Structure and content in language production: A theory of frame constraints in phonological speech errors', *Cognitive Science* **17**, 145–195.

Dupoux, Emmanuel, Christophe Pallier, Kazuhiko Kakehi & Jacques Mehler (2001), 'New evidence for prelexical phonological processing in word recognition', *Language and Cognitive Processes* **5**(16), 491–505.

Elman, Jeffrey L. (1988), Finding structure in time, Technical Report 9901, Center for Research in Language, UCSD, CA.

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

Gasser, Michael (1992), Learning distributed representations for syllables, *in* 'Proc. of 14th Annual Conference of Cognitive Science Society', pp. 396–401.

Haykin, Simon (1994), *Neural Networks*, Macmillian Publ, NJ.

Kaplan, Ronald & Martin Kay (1994), 'Regular models of phonological rule systems', *Computational Linguistics* **20/3**, 331–378.

Kessler, Brett & Rebecca Treiman (1997), 'Syllable structure and the distribution of phonemes in english syllables', *Journal of Memory and Language* **37**, 295–311.

Konstantopoulos, Stasinos (2003), Using Inductive Logic Programming to Learn Local Linguistic Structures, PhD thesis, Rijksuniversiteit Groningen.

Kuan, Chung-Ming, Kurt Hornik & Halbert White (1994), 'A convergence result for learning in recurrent neural networks', *Neural Computations* **6**, 420–440.

Laver, John (1994), *Principles of Phonetics*, Cambridge Univeristy Press, Cambridge.

Lawrence, Steve, C. Lee Giles & S. Fong (1995), On the applicability of neural networks and machine learning methodologies to natural language processing, Technical report, Univ. of Maryland.

Luce, Paul L., David B. Pisoni & Steven D. Goldinger (1990), Similarity neighborhoods of spoken words, *in* G. T. M. Altmann, ed., 'Cognitive Models of Speech Processing', A Bradford Book, Cambridge, Massachusetts, USA, pp. 122–147.

McQueen, James (1998), 'Segmentation of continuous speech using phonotactics', *Journal of Memory and Language* **39**, 21–46.

Mitchell, Thomas (1997), *Machine Learning*, McGraw Hill College.

Nerbonne, John, Wilbert Heeringa & Peter Kleiweg (1999), Edit distance and dialect proximity, *in* D. Sankoff & J. Kruskal, eds, 'Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison, 2nd ed.', CSLI, Stanford, CA, pp. v–xv.

Norris, D., J.M. McQueen, A. Cutler & S. Butterfield (1997), 'The possible-word constraint in the segmentation of continous speech', *Cognitive Psychology* **34**, 191–243.

Omlin, Christian W. & C. Lee Giles (1996), 'Constructing deterministic finite-state automaya in recurrent neural networks', *Journal of the ACM* **43(6)**, 937–972.

Pacton, S., P. Perruchet, M. Fayol & A. Cleeremans (2001), 'Implicit learning in real world context: The case of orthographic regularities', *Journal of Experimental Psychology: General* **130(3)**, 401–426.

Plaut, D.C., J. McClelland, M. Seidenberg & K. Patterson (1996), 'Understanding normal and impaired word reading: Computational principles in quasi-regular domains', *Psychological Review* **103**, 56–115.

Reed, Russell D. & Robert J. Marks II (1999), *Neural Smithing*, MIT Press, Cambridge, MA.

Reilly, Ronan (2002), 'The relationship between object manipulation and language development in broca's area: A connectionist simulation of greenfield's hypothesis', *Behavioral and Brain Sciences* **25**, 145–153.

Robinson, A. J. & F. Fallside (1988), Static and dynamic error propagation networks with application to speech coding, *in* D. Z. Anderson, ed., 'Neural Information Processing Systems', American Institute of Physics, NY.

Rodd, Jenifer (1997), Recurrent neural-network learning of phonological regularities in Turkish, *in* 'Proc. of Int. Conf. on Computational Natural Language Learning', Madrid, pp. 97–106.

Rumelhart, David E. & James A. McClelland (1986), *Parallel Distributed Processing - Explorations of the Microstructure of Cognition*, The MIT Press, Cambridge, MA.

Rumelhart, D.E., G.E. Hinton & R.J. Williams (1986), Learning internal representations by error propagation, *in* D. E. Rumelhart & J. A. McClelland, eds, 'Parallel Distributed Processing - Explorations of the Microstructure of Cognition, Volume 1, Foundations', The MIT Press, Cambridge, MA, pp. 318 – 363.

Shillcock, R., Paul Cairns, Nick Chater & Joe Levy (1997), Statistical and connectionist modelling of the development of speech segmentation, *in* Broeder & Murre, eds, 'Models of Language Learning', MIT Press.

Shillcock, Richard, Joe Levy, Geoff Lindsey, Paul Cairns & Nick Chater (1993), Connectionist modelling of phonological space, *in* T. M. Ellison & J. Scobbie, eds, 'Computational Phonology', Edinburgh, pp. 179–195. Edinburgh Working Papers in Cognitive Science 8.

Stoianov, Ivelin Peev (1998), Tree-based analysis of simple recurrent network learning, *in* '36 Annual Meeting of the Association for Computational Linguistics and 17 Int. Conference on Compuational Linguistics', Vol. 2, Montreal, Canada, pp. 1502–1504.

Stoianov, Ivelin Peev (2001), Connectionist Lexical Modellinig, PhD thesis, University of Groningen, the Netherlands.

Stoianov, Ivelin Peev & John Nerbonne (2000), Exploring phonotactics with simple recurrent networks, *in* F. van Eynde, I. Schuurman & N. Schelkens, eds, 'Computational Linguistics in the Netherlands, 1998', Rodopi, Amsterdam, NL, pp. 51–68.

Stoianov, Ivelin Peev, John Nerbonne & Huub Bouma (1998), Modelling the phonotactic structure of natural language words with simple recurrent networks, *in* P.-A. Coppen, H. van Halteren & L. Teunissen, eds, 'Computational Linguistics in the Netherlands, 1997', Rodopi, Amsterdam, NL, pp. 77–96.

Stoianov, Ivelin Peev, Laurie Stowe & John Nerbonne (1999), Connectionist learning to read aloud and correlation to human data, *in* '21 Annual Meeting of the Cognitive Science Society, Vancouver, Canada', Lawrence Eurlbaum Ass., London, pp. 706–711.

Stowe, Laurie, Anton Wijers, A. Willemsen, Eric Reuland, A. Paans & Wim Vaalburg (1994), 'Pet studies of language: An assessment of the reliability of the technique', *Journal of Psycholinguistic Research* **23**(6), 499–527.

Tjong Kim Sang, Erick (1995), The limitations of modeling finite state grammars with simple recurrent networks, *in* 'Proc of 5-th CLIN', pp. 133–143.

Tjong Kim Sang, Erick (1998), Machine Learning of Phonotactics, PhD thesis, University of Groningen.

Tjong Kim Sang, Erik & John Nerbonne (1999), Learning simple phonotactics, *in* 'Proceedings of the Workshop on Neural, Symbolic, and Reinforcement Methods for Sequence Processing, ML2 workshop at IJCAI'99', pp. 41–46.

Treiman, R. & A. Zukowski (1990), 'Toward an understanding of english syllabification', *Journal of Memory and Language* **34**, 66–85.

Tsoi, Ah Chung & Andrew Back (1997), 'Discrete time recurrent neural network architectures: A unifying revew', *Neurocomputing* **15**, 183–223.