Connection Science
Vol. 18, No. 3, September 2006, 287–302

Taylor & Francis
Taylor & Francis Group

Learn more by training less: systematicity in sentence processing by recurrent networks

STEFAN L. FRANK*

Nijmegen Institute for Cognition and Information, Raboud University Nijmegen,
PO Box 9104, 6500 HE Nijmegen, The Netherlands

(Received 28 November 2005; revised version received 21 March 2006; accepted 23 April 2006)

Connectionist models of sentence processing must learn to behave systematically by generalizing from a small training set. To what extent recurrent neural networks manage this generalization task is investigated. In contrast to Van der Velde *et al.* (*Connection Sci.*, 16, pp. 21–46, 2004), it is found that simple recurrent networks do show so-called weak combinatorial systematicity, although their performance remains limited. It is argued that these limitations arise from overfitting in large networks. Generalization can be improved by increasing the size of the recurrent layer without training its connections, thereby combining a large short-term memory with a small long-term memory capacity. Performance can be improved further by increasing the number of word types in the training set.

Keywords: Sentence processing; Systematicity; Generalization; Overfitting; Simple recurrent network; Echo state network

1. Introduction

An important property of human language is that innumerable sentences can be constructed using only a limited number of words and ways to combine them. Of the different mechanisms that give rise to this property, recursiveness has traditionally received a lot of attention in linguistic research. It arises because one of the constituents of a sentence can itself be a sentence, possibly resulting in infinite levels of recursion and, therefore, an infinite number of possible sentences. However, since people cannot actually deal with more than two or three levels of embedded sentences, *combinatorial systematicity* may be more important than recursiveness.

Combinatorial systematicity arises when a limited lexicon is combined with a limited number of sentence structures, resulting in a very large (but finite) number of possible sentences.[†] For instance, combining the very simple sentence structure N-V-N ('noun verb

*Email: S.Frank@nici.ru.nl

[†]Van der Velde *et al.* (2004) referred to this phenomenon as 'combinatorial *productivity*'. According to Fodor and Pylyshyn (1988: 33), the difference between systematicity and productivity is that productivity results in 'unbounded expressive power'. Since combinatorial systematicity does not give rise to an unbounded number of possible sentences, it is not productive in this sense. Therefore, the term 'systematicity' will be used in this paper.

Connection Science
ISSN 0954-0091 print/ISSN 1360-0494 online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/09540090600768336

noun') with m different nouns and the same number of verbs, results in m^3 possible sentences. Since most adults know several thousand content words, the number of possible N-V-N sentences can be in the order of 10^{10}. Although most of these will not make any sense, people can understand *all* of them in the sense that they are able to answer 'who did what to whom?' questions about the sentence content, as argued by Van der Velde *et al.* (2004). They gave *houses eat furniture* as an example of a nonsense sentence that people can nevertheless process well enough to answer correctly the question 'what eats furniture?' According to Van der Velde *et al.*, human language users must rely on combinatorial systematicity to perform such a task.

To be considered as candidates for cognitive models, neural networks must be able to display (or, better still, to explain) the systematicity present in human language. In this context, their potential has been fiercely debated (e.g. Fodor and Pylyshyn 1988, Chalmers 1993, Aizawa 2003). The extent to which particular connectionist models of language processing show systematicity was investigated by Hadley (1994), who noted that systematic behaviour is basically an issue of learning and generalization: neural networks are trained on a limited number of sentences from which they should generalize to be able to process all possible sentences. Moreover, since people learn systematic language behaviour from experiencing only a small fraction of possible sentences, a neural network should be trained on relatively few sentences if it is to form a realistic model of this behaviour. Next, Hadley made a distinction between weak and strong systematicity. A network is weakly systematic if it can process sentences that have novel combinations of words, but these words are in the syntactic positions they also occurred in during training. For instance, a network shows weak systematicity if it is trained on *boy sees girl* and *dog chases cat*, and can generalize to *boy chases cat*. This network did not need to process any word occurring in a new syntactic position, because in the training sentences as well as in the test sentence, *boy* is subject and *cat* is direct object. Strong systematicity, on the other hand, does require generalization to new syntactic positions. For instance, if a network is to show strong systematicity, it has to be able to process *girl sees boy* without being trained on any sentence in which *girl* is subject or *boy* is direct object. Moreover, to satisfy Hadley's definition of strong systematicity, the network must also be weakly systematic and be able to process sentences with embedded clauses containing words in new syntactic positions.

According to Hadley (1994), the connectionist models of that time were weakly systematic at best. Of the connectionist demonstrations of strong systematicity that have appeared since, some relied on representations or network architectures that were tailored specifically for obtaining systematicity (Hadley and Hayward 1997, Hadley and Cardei 1999, Hadley *et al.* 2001, Bodén 2004). Others were restricted to just a single test item (Elman 1998) or reported mixed results (Christiansen and Chater 1994). All in all, the ability of neural networks to display robust, scalable, strong systematicity has not been demonstrated convincingly.

Van der Velde *et al.* (2004) claimed that even *weak* combinatorial systematicity generally lies beyond the abilities of the most popular connectionist system for sentence processing: the simple recurrent network (SRN; Elman 1990). They constructed a simple language with just eight nouns and eight verbs, as well as the relative pronoun *who* and an end-of-sentence marker (which will be denoted by *[end]* and also be considered a word). Crucially, the 16 content words were divided into four groups of two nouns and two verbs each. For instance, one group contained the content words *boy, girl, sees* and *hears*, while another contained *dog, cat, chases* and *kicks*. In the sentences used for training, all content words came from the same group, so *boy sees girl [end]* and *dog chases cat [end]* were in the training set, but *boy chases cat [end]* was not. This set-up ensured that the proportion of sentences used for training remained relatively small (less than 0.44%).

An SRN was presented with the training sentences, one word at a time, and was trained to predict which word would come next, as is a common task for this type of network. If the

network is to show weak combinatorial systematicity after training, it should be able also to process sentences it was not trained on (such as *boy chases cat*), in which content words come from different groups. To perform the word-prediction task on these novel sentences, it would need to differentiate between nouns and verbs, as well as implicitly learn the N-V-N sentence structure the words can be put into. Contrary to this, Van der Velde *et al.* (2004) found that the network performed much worse on novel sentences than it did on training sentences. It was concluded that SRNs cannot generalize from a small set of training sentences to show weak combinatorial systematicity. Therefore, they cannot scale up to more comprehensive languages and do not form a realistic model of human linguistic performance.

This conclusion, however, may be premature. First, it is not certain that Van der Velde *et al.*'s (2004) SRN did not behave systematically at all. They claimed that 'the networks failed when words from [training] sentences ... were combined' (p. 40), but what it means for the network to 'fail' is not defined. It clearly did worse on test sentences than on training sentences, but may still have performed better than a network that behaves completely unsystematically.

Second, Van der Velde *et al.* (2004) did not investigate whether a network of a different size may generalize better. The network that was used could have been too small to handle many different sentences, or, alternatively, it could have been too large to show generalization. In that case, a phenomenon known as *overfitting* occurs: the network has so many free parameters (i.e. trainable connection weights) relative to the size of the training set that all training examples can simply be individually stored, resulting in impoverished generalization (see, e.g. Baum and Haussler 1989, Müller *et al.* 1996).

Third, the language that was used may have been too simple. A language with more different content words may make it more useful for the network to make (implicity) a distinction between nouns and verbs, rather than storing only occurrences of individual words. Such abstraction is necessary for systematicity to occur.

This paper reinvestigates the extent to which recurrent networks show weak combinatorial systematicity. It takes Van der Velde *et al.* (2004) as a starting point, but deals with the three points mentioned above by applying a more sophisticated measure of network performance, varying network size and varying language size. It will be shown that, in contrast to Van der Velde *et al.*'s claim, SRNs can handle weak combinatorial systematicity to some extent. Although using a larger lexicon indeed increases the network's ability to generalize, this ability remains limited. Presumably this is because small networks lack the required processing capacity, while larger networks suffer from overfitting. It will be discussed that this problem can (at least partially) be solved by reducing the number of trainable connections while increasing network size, which is accomplished by *not training* the recurrent connections of the SRN. This turns the SRN into a fairly new type of recurrent network, the so-called echo state network (ESN; Jaeger 2003, Jaeger and Haas 2004), which will be shown to handle weak combinatorial systematicity better than does the SRN.

2. Set-up of simulations

2.1 *Training and test sentences*

The training and test sentences were constructed using the grammar in table 1, which subsumes the grammar used by Van der Velde *et al.* (2004). The language consists of simple sentences, which have an N-V-N structure, and two types of complex sentences, which contain one relative clause. Complex sentences can be either right-branching (N-V-N-*who*-V-N) or centre-embedded (N-*who*-N-V-V-N). English equivalents of right-branching and centre-embedded

Table 1. Grammar for producing training and test sentences.[†]

S	→	Simple | Right | Centre
Simple	→	N V N *[end]*
Right	→	N V N *who* V N *[end]*
Centre	→	N *who* N V V N *[end]*
N	→	$N_1 \mid N_2 \mid N_3 \mid N_4$
V	→	$V_1 \mid V_2 \mid V_3 \mid V_4$
N_x	→	$n_{x1} \mid n_{x2} \mid \ldots$
V_x	→	$v_{x1} \mid v_{x2} \mid \ldots$

[†]Nouns (n) and verbs (v) are divided into four groups (x denotes group number). The number of nouns and verbs in each group is varied from two to five. In every training sentence, all content words come from the same group.

sentences would be, for instance, *boy sees girl who chases cat* and *girl who boy sees chases cat*, respectively.

Content words (nouns and verbs) are divided into four groups: $N_1, \ldots, N_4$ and $V_1, \ldots, V_4$. Nouns from group x are labelled $n_{x1}, n_{x2}, \ldots$ and verbs from group x are labelled $v_{x1}, v_{x2}, \ldots$ (only abstract labels are used to refer to content words). The number of different content words per group is varied: If w denotes the lexicon size (i.e. the total number of word types in the language), each group has $(w-2)/8$ nouns and the same number of verbs. For $w = 18$ (its minimum value) there are four content words per group. In that case, the language is identical to the one used by Van der Velde *et al.* (2004), apart from the irrelevant detail that they used actual English content words instead of abstract labels. The largest lexicon applied here has $w = 42$ words, making 10 content words per group.

The training set consisted of all sentences in which all content words were taken from the same group. This means that simple training sentences had the form '$n_{xi}\ v_{xj}\ n_{xk}$ *[end]*', and that right-branching and centre-embedded training sentences had the form '$n_{xi}\ v_{xj}\ n_{xk}$ *who* $v_{xl}\ n_{xm}$ *[end]*' and 'n_{xi} *who* $n_{xj}\ v_{xk}\ v_{xl}\ n_{xm}$ *[end]*', respectively. Here, the range of the indices $i, \ldots, m$ depends on the lexicon size: $i, j, k, l, m \in \{1, \ldots, (w-2)/8\}$.

In contrast to the training sentences, each test sentence contained content words from different groups. Van der Velde *et al.* (2004) showed that the most difficult test sentences for the network to process were those containing content words from as many different groups as possible. To perform the strictest possible test for weak combinatorial systematicity, these difficult test sentences were used here. That is, all simple test sentences were of the form '$n_{xi}\ v_{yj}\ n_{zk}$ *[end]*', where $x \neq y \neq z$, and all such sentences were used for testing. Likewise, the five content words in right-branching and centre-embedded test sentences came from all four different groups. Two complex test sets were constructed randomly, one containing 100 right-branching sentences and the other containing 100 centre-embedded sentences.

Table 2 gives the sizes of the training and test sets as a function of lexicon size w. Also, the table shows what fraction of all possible sentences is used for training. Note that this fraction is smaller for larger lexicons.

2.2 *Training the networks*

The networks processed the sentences one word at a time. After each input word, they had to predict which word would come next. All sentences were concatenated (in random order) into one input stream, so the networks also had to predict the word following *[end]*, that is, the next sentence's first word.

Table 2. Numbers of sentence types in training and test sets as a function of the lexicon size w, and fraction of possible sentences used for training.

	Training set size		Test set size		
w	Simple	Complex	Simple	Complex	Fraction Trained ($\times 10^{-3}$)
18	32	256	192	200	4.36
26	108	1944	648	200	4.11
34	256	8192	1536	200	4.02
42	500	25 000	3000	200	3.98

Following Elman (1991, 1993), Van der Velde *et al.* (2004) first trained their network on simple sentences only, after which the ratio of complex to simple training sentence tokens was gradually increased to 4 : 1. Rhode and Plaut (1999) demonstrated that such a scheme is not needed for successful SRN training. Therefore, in the experiments described here, the 4 : 1 ratio of complex to simple sentences was present from the start of training.

Networks were trained for 10 epochs, during each of which 31 250 training sentence tokens were presented. With lexicon size $w = 42$, this comes down to the 25 000 different complex training sentences and $25\,000/4 = 6250$ simple sentences. If $w < 42$, each sentence was presented more than once in each epoch so that the number of sentence tokens per epoch remained 31 250. This very large number of training inputs ensured that the networks learned to process the training sentences (nearly) perfectly, so that any effect of lexicon or network size on generalization could not be attributed to general differences in network performance.

Initial connection weight settings were chosen randomly, uniformly distributed between -0.15 and $+0.15$. For each lexicon and network size, 10 networks were trained, differing only in their random initial weight setting. All results presented hereafter are averaged over the results of these 10 networks. The backpropagation algorithm (without momentum) was used for weight updating, which occurred after every processed word. The learning rate, set to 0.1 initially, was decreased in steps of 0.01 after each epoch. Cross-entropy was used as the error function, and output units had 'softmax' activation functions:

$$a_{\text{out},i} = f_{\text{out}}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}, \tag{1}$$

where x_i is the total activation received by output unit i and $a_{\text{out},i}$ is its resulting output activation. Applying this function results in output activations that sum to one and can be interpreted as probabilities. Since a localist representation was used, output unit i represents word i, so $a_{\text{out},i}$ is the network's estimated probability that the next input will be word i.

2.3 *Rating performance and systematicity*

There exist several measures for rating network performance in the word-prediction task. Some require the network's output to reflect the precise probabilities that each word will be the next input (e.g. Elman 1990, 1991, 1993, Rohde and Plaut 1999, Bodén 2004), while others consider an output as correct when the only activated units are those representing words that can be the next input (e.g. Christiansen and Chater 1999, MacDonald and Christiansen 2002, Van der Velde *et al.* 2004). None of these measures are suitable for investigating whether a network generalizes well enough to display combinatorial systematicity because they lack a baseline: a specific level of performance above which the network can be claimed to process a novel input better than would a non-generalizing network. Below, a performance measure is defined that does have such a baseline.

Let G be the set of words that form grammatical continuations of the input so far. The performance measure assigns the maximum performance score of 1 to any network output that is flawless, meaning that only output units representing words in G are activated. That is, an output vector $\mathbf{a}_{\text{out}}$ is flawless if and only if $a_{\text{out},i} = 0$ for all $i \notin G$. In that case, the total 'grammatical' output activation $a(G) = \sum_{i \in G} a_{\text{out},i} = 1$. Conversely, the minimum performance score of -1 is assigned to an output that is completely incorrect, that is, when $a_{\text{out},i} = 0$ for all $i \in G$, making $a(G) = 0$.[†]

The performance score equals the baseline value of zero for any output of a hypothetical network that processes the training sentences perfectly but does not generalize *at all*. When confronted with a test sentence, such a network cannot combine the information from different words in the input because that would require generalization to this new input. Instead, the best it can do is to use the last word of the sequence to base its next-word prediction on, resulting in the same behaviour as a perfectly trained, non-recurrent, feedforward network. By definition, each output activation $a_{\text{out},i}$ of this network always equals the probability that word i will come next *given only the current word*. This probability, denoted p_i, is estimated from the training sentences. Note that p_i generally differs from the actual probability that i is the next word, which depends on *all* previous words of the sentence, not only on the current word.

As an example, suppose the input '*[end]* n_{11} v_{22}' is processed. The *[end]* symbol indicates that the following two words form the beginning of a sentence. This is a test sentence because the words come from different groups. According to the grammar of table 1, the next input can be any noun and nothing else. Therefore, the set G of grammatical continuations contains all nouns and no other words. However, the non-generalizing network cannot handle the novel input. All it can do is make use of only the current input word v_{22}. In fact, Van der Velde *et al.* (2004) provided some evidence that their network did behave according to these 'word–word associations', as they called them. During training, the words that followed v_{22} were the nouns from group 2 (when v_{22} occurred in simple or right-branching sentences) and the verbs from group 2 (when v_{22} occurred as the fourth word of a centre-embedded sentence). This means that p_i, the probability that word i follows v_{22} in the training set, equals zero for all words except the nouns and verbs from group 2.

The probability of correctly predicting a word in G from the current word only equals $p(G) = \sum_{i \in G} p_i$. By definition, the hypothetical, non-generalizing network always results in $a_{\text{out},i} = p_i$, so $a(G) = p(G)$. A network that does better than this will have more activation end up in units representing words in G. Consequently, $a(G) > p(G)$, resulting in a positive performance score. A network that does worse results in $a(G) < p(G)$ and a negative score.

To summarize, the performance score is $+1$ if $a(G) = 1$, it is -1 if $a(G) = 0$, and it is zero whenever $a(G) = p(G)$. This is expressed formally by:

$$\text{performance} = \begin{cases} \dfrac{a(G) - p(G)}{p(G)} & \text{if } a(G) \leq p(G) \\ \dfrac{a(G) - p(G)}{1 - p(G)} & \text{if } a(G) > p(G). \end{cases} \tag{2}$$

Note that this scheme fails when $p(G) = 1$, which happens when the set G of grammatically correct next words depends only on the current word. In such an event, generalization is not required for making a flawless prediction, and even a perfect output (i.e. $a(G) = 1$) would result in a performance of zero. In practice, this happens only at the end-of-sentence marker *[end]*, when the first word of the next sentence has to be predicted. At this point, the next word

[†]Note that, unless G contains all words, $a(G)$ can only be infinitesimally close to one since $a(G) = 1$ would require some e^{x_i} (of equation (1)) to equal zero, which can only happen if connection weights are infinite. For the same reason, $a(G) = 0$ is impossible unless $G = \emptyset$. This also means that a performance score of exactly ± 1 is unattainable.

must be a noun, regardless of the inputs preceding *[end]*. Therefore, network performance when processing *[end]* remains undefined.

3. Simple recurrent network simulations

3.1 *The simple recurrent network*

The SRN used for the experiments described here consists of four layers of units, as shown in figure 1. These are called the input, recurrent, hidden and output layer, respectively. Input and output layers have w units, one for each word in the lexicon. The number of units in the recurrent layer, denoted n, is varied to investigate the effect of network size. The hidden layer has 10 units, as was the case for the corresponding layer in Van der Velde *et al.*'s (2004) network. That network differed from the SRN used here because it included a fifth layer in-between the input and recurrent layers. In the network of figure 1, this layer was left out to make the network isomorphic to the ESN of section 4.1.

The network processed the input sequence over a number of processing time steps, receiving one word per time step. The input at time step t is encoded by the input activation column vector $\mathbf{a}_{\text{in}}(t) = (a_{\text{in},1}(t), \ldots, a_{\text{in},w}(t))'$, where $a_{\text{in},i}(t) = 1$ for input word i and all other vector elements are 0 (i.e. the encoding is localist). This activation is sent via the recurrent layer (where it is combined with that layer's activation at the previous time step) and the hidden layer to the output units, according to

$$\begin{aligned}
\mathbf{a}_{\text{rec}}(t) &= \mathbf{f}\left(\mathbf{W}_{\text{in}}\mathbf{a}_{\text{in}}(t) + \mathbf{W}_{\text{rec}}\mathbf{a}_{\text{rec}}(t-1) + \mathbf{b}_{\text{rec}}\right) \\
\mathbf{a}_{\text{hid}}(t) &= \mathbf{f}\left(\mathbf{W}_{\text{hid}}\mathbf{a}_{\text{rec}}(t) + \mathbf{b}_{\text{hid}}\right) \\
\mathbf{a}_{\text{out}}(t) &= \mathbf{f}_{\text{out}}\left(\mathbf{W}_{\text{out}}\mathbf{a}_{\text{hid}}(t) + \mathbf{b}_{\text{out}}\right),
\end{aligned} \tag{3}$$

where: $\mathbf{a}_{\text{rec}}(t), \mathbf{a}_{\text{hid}}(t), \mathbf{a}_{\text{out}}(t)$ are the activation vectors at time step t of the recurrent, hidden and output layers, respectively; $\mathbf{b}_{\text{rec}}, \mathbf{b}_{\text{hid}}, \mathbf{b}_{\text{out}}$ are the corresponding bias vectors; $\mathbf{W}_{\text{in}}, \mathbf{W}_{\text{rec}}, \mathbf{W}_{\text{hid}}, \mathbf{W}_{\text{out}}$ are connection weight matrices; $\mathbf{f}$ is the logistic activation function; and $\mathbf{f}_{\text{out}}$ is the softmax activation function of equation (1). Each element i of $\mathbf{a}_{\text{out}}(t)$ is the network's estimated probability that word i will be the input at time step $t + 1$.

All values in all connection-weight matrices $\mathbf{W}$ and bias vectors $\mathbf{b}$ are set during network training, as described in section 2.2.

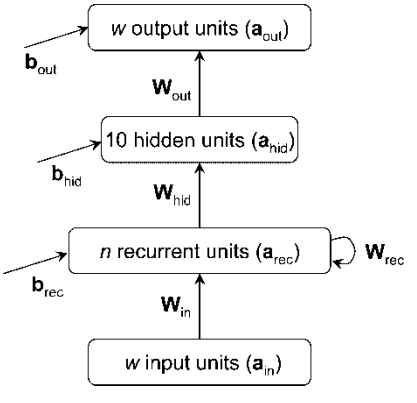

Figure 1. Architecture of the four-layer SRN. Vectors **a** refer to activations of units, **W** are connection-weight matrices and **b** are bias vectors.

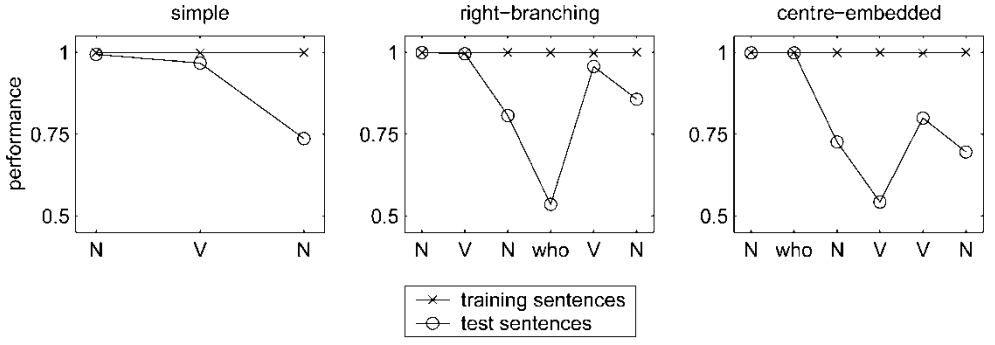

Figure 2. SRN performance at each word of simple, right-branching and centre-embedded training and test sentences ($n = 20$; $w = 18$).

3.2 *Results*

First, results are presented for the network with recurrent layer size $n = 20$, trained on the language with lexicon size $w = 18$. These are the same values as used by Van der Velde *et al.* (2004). As indicated in figure 2 by the high performance scores, the training sentences were learned almost perfectly. At any word of any sentence type, the performance (averaged over all training sentences) was above 0.997. Performance on test sentences, however, was much lower. Especially on the third word of simple sentences and the fourth word of complex sentences, the networks were not very good at predicting which words could follow.

From similar results, Van der Velde *et al.* (2004) concluded that SRNs do not handle weak combinatorial systematicity. However, even at the words for which performance on test sentences is minimal, it is still significantly above zero, as indicated by sign tests (for simple sentences: $z = 33.0$, $p < 10^{-16}$; right-branching: $z = 17.2$, $p < 10^{-16}$; centre-embedded: $z = 18.9$, $p < 10^{-16}$).

Strictly speaking, these results do not form proof of systematicity by the network. It is possible that nearly all test sentences are processed incorrectly, that is, that at least one word of every sentence results in a negative performance score. If this is not always the same word, averaging over test sentences can result in the purely positive scores of figure 2. However, this turned out not to be the case. Of the 200 complex test sentences processed by the 10 networks, 61.7% were processed correctly, that is, resulted in positive performance at each word.[†]

Next, it was investigated whether varying recurrent layer size n and lexicon size w can bring performance on test sentences into the range of training sentence performance. Only averages over all words will be presented, because the general pattern of performance scores over words was similar for all network and lexicon sizes. This can be seen in figure 3, which shows the performance scores for those combinations of values $n \in \{10, 20, 40, 70, 100\}$ and $w \in \{18, 26, 34, 42\}$ that resulted in the best and the worst results, as well as averaged over all levels of n and w. Although performance varies strongly with n and w, the pattern of performance scores over words does not depend much on the general level of performance. For instance, it is always the third and fourth words that result in the lowest performance scores.

Figure 4 shows the effect of recurrent layer size. Although performance on test sentences (averaged over all four lexicon sizes) improves as n increases from 10 to 40, it decreases as networks are made even larger. In general, the optimal recurrent layer size seems to be around

[†]Simple test sentences can safely be ignored here because the first three words of right-branching sentences form a simple sentence themselves. Therefore, if simple sentences were often processed incorrectly, this would also show up in the performance scores of right-branching sentences.

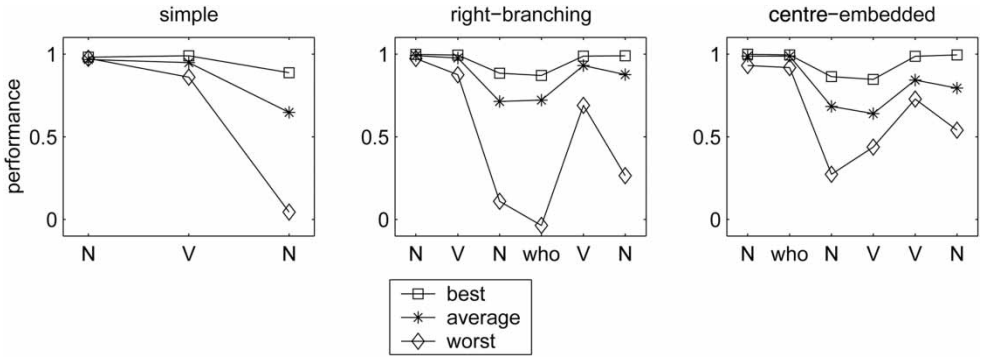

Figure 3. SRN performance, averaged over test sentences, for best and worst levels of n and w, as well as averaged over all levels of n and w.

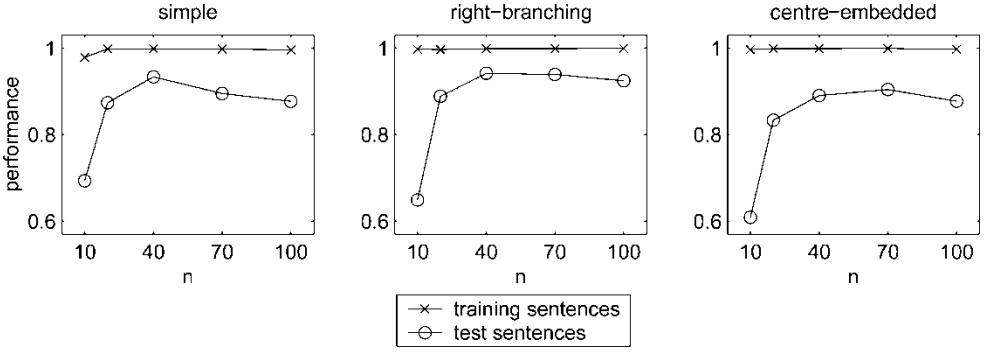

Figure 4. SRN performance as a function of network size n, averaged over all words and lexicon sizes of simple, right-branching and centre-embedded training and test sentences.

$n = 40$. Wilcoxon match-pairs signed ranks tests indicated that performance on test sentences is significantly lower for $n = 100$ than for $n = 40$ (for simple sentences: $z = 9.98$, $p < 10^{-16}$; right-branching: $z = 5.64$, $p < 10^{-7}$; centre-embedded: $z = 9.18$, $p < 10^{-16}$).

Figure 5 shows the effect of varying lexicon size w on the performance of the SRN with $n = 40$. Making use of the fact that performance scores (or, rather, one minus these scores) are approximately gamma-distributed, a general linear model was fitted. This analysis revealed

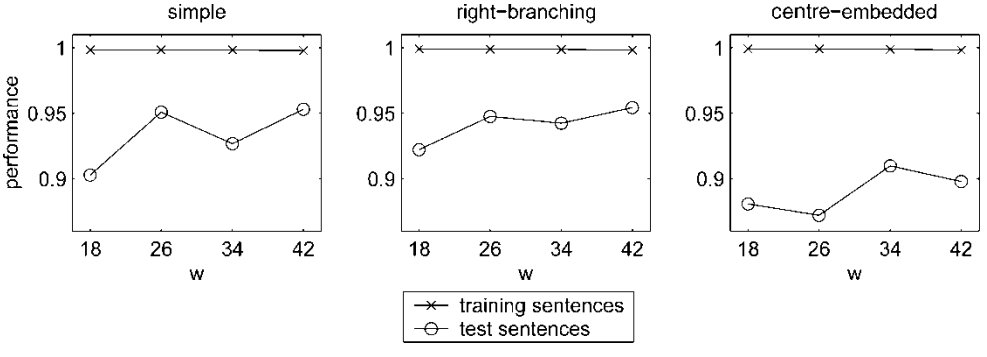

Figure 5. SRN performance as a function of lexicon size, averaged over all words of simple, right-branching and centre-embedded training and test sentences ($n = 40$).

that increasing the number of different words improves generalization. Although the effect is quite small, there was a significant, positive, linear effect of lexicon size w on performance score for test sentences (for simple sentences: $z = 13.0$, $p < 10^{-16}$; right-branching: $z = 5.62$, $p < 10^{-7}$; centre-embedded: $z = 3.98$, $p < 10^{-4}$). With $n = 40$ and $w = 42$, 84.2% of complex test sentences were processed correctly.

3.3 *Conclusion*

Even a small SRN ($n = 20$) being trained on a small lexicon ($w = 18$) learns to generalize to test sentences. On each word of test sentences, average performance was above zero. Also, a majority of complex test sentences was processed correctly. It can be concluded that an SRN can display some, but by no means perfect, weak combinatorial systematicity. By increasing lexicon and recurrent layer size, both the average performance scores and the percentage of correctly processed complex test sentences could be increased.

Recurrent layer size had a non-monotonic effect on generalization, with medium sized networks leading to the best performance on test sentences. The reason for this could be that small networks lack the capacity to perform the task while very large networks have so many trainable connection weights that overfitting occurs. In that case, the networks learn to process the training sentences perfectly by storing each training example in the weight matrices, without any need to generalize. If this indeed explains the non-monotonic effect in figure 4, generalization might be improved further by using a large network that nevertheless has few trainable weights. The ESN discussed next is just such a network.

4. Echo state network simulations

4.1 *The echo state network*

Christiansen and Chater (1999) studied the ability of SRNs to process different types of recursive sentence structures and found that even *before training* these networks were sensitive to structural differences among the sentences. This phenomenon was investigated further by Tiňo *et al.* (2002, 2004), who managed to perform next-symbol prediction on language-like input by extracting information from the recurrent layer of an untrained SRN. Although a trained network resulted in more accurate predictions, such findings do raise the question to what extent training the recurrent connections is necessary. Indeed, independently from each other, Jaeger (2003) and Maass *et al.* (2002) proposed a new type of recurrent neural network based on the idea that the recurrent, dynamical part of the network does not require training. Instead, it serves as a non-specific memory, the so-called dynamical reservoir, that retains information about the input sequence by letting activation 'bounce around' the recurrent units. A separate 'readout' network is trained to interpret the dynamical reservoir's activation states.

Such networks are called echo state networks (ESN; Jaeger) or liquid state machines (LSM; Maass *et al.*). Conceptually, ESNs and LSMs are similar systems.[†] The network used here is an adaptation of the ESNs presented by Jaeger (2003) and Jaeger and Haas (2004). Since those networks did not have a hidden layer between the dynamical reservoir and the readout units, they could be trained very quickly without requiring backpropagation or any other incremental weight-updating algorithm. However, preliminary tests (reported in Appendix A) revealed that

[†]In practice, ESNs use continuously valued processing units while LSMs model spiking neurons. For this reason, LSMs are more often used for simulating biological neural networks, while ESNs are more convenient for the cognitive simulations discussed here.

such an additional hidden layer helps the ESN to perform as well as an SRN on the word-prediction task. Also, Jaeger's (2003) ESN included direct connections from the input to the output units, while Jaeger and Haas's network had feedback connections from the output units back into the dynamical reservoir. To make the ESN used here as similar as possible to the SRN of figure 1, such input-to-output and feedback connections were not included. As a result, the ESN performs the same computations as the SRN, except that equation (3), computing the recurrent layer's activations, is replaced by

$$\mathbf{a}_{\text{rec}}(t) = \mathbf{W}_{\text{in}}\mathbf{a}_{\text{in}}(t) + \mathbf{W}_{\text{rec}}\mathbf{a}_{\text{rec}}(t-1).$$

Note that the ESN's recurrent layer (i.e. its dynamical reservoir), unlike the SRN's, has no bias vector and that its units have no activation function, that is, they are linear. Although it is more common to use sigmoidal units in the dynamical reservoir, preliminary investigations (see Appendix A) resulted in much better performance with linear units. As will be discussed in section 5.1, Jaeger (2002) reported a similar finding.

An ESN's connection-weight matrices $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_{\text{rec}}$ are not trained. Instead, they keep their initial random values. The weight matrices $\mathbf{W}_{\text{hid}}$ and $\mathbf{W}_{\text{out}}$, as well as the two remaining bias vectors, are trained in the same way as their SRN counterparts, as described in section 2.2.

Obviously, the number of trainable connections of an ESN is much smaller than that of an SRN the same size. Moreover, this number increases quadratically with recurrent layer size in an SRN, but only linearly in an ESN. With lexicon size $w = 42$, the SRN of figure 1 has $n^2 + 53n + 472$ parameters (connection weights and biases) to set. For the ESN, this number is only $10n + 472$. This makes the ESN faster to train than the SRN, but more importantly, the ESN might show better generalization if it is indeed the case that very large SRNs generalize poorly because they have enough free parameters simply to memorize all training sentences. Increasing the recurrent layer size n of an ESN could offer the advantage of larger processing capacity without the drawback of worse generalization. Here, this idea is tested empirically.

Although the recurrent weights are chosen randomly, there are three constraints they must meet in order to be useful. First, the recurrent layer has to be large enough. Here, its size is varied between $n = 10$ and $n = 100$. Second, it has to be sparsely connected, that is, only a small percentage of possible connections should be present. This is because sparse connectivity results in loosely connected 'islands' with different dynamical properties, some of which will turn out to be useful for learning the task at hand. In the ESN used here, only 15% of recurrent connection weights are non-zero. Third, the spectral radius (i.e. the largest absolute eigenvalue) of $\mathbf{W}_{\text{rec}}$ should have a value that matches the 'speed of the task'. A spectral radius close to 1 results in a network that produces slow-changing dynamics, while smaller spectral radii result in faster networks.[†] This value can be set by generating a random weight matrix, dividing all weights by the spectral radius of the matrix (resulting in a spectral radius of 1) and then multiplying the weights by the intended spectral radius. The matrix $\mathbf{W}_{\text{rec}}$ used here has a spectral radius of 0.7.

4.2 *Results and conclusion*

The performance scores of the ESN with $n = 20$ recurrent units and lexicon size $w = 18$ are shown in figure 6. Although the pattern of performance over words is similar to what was found with an SRN of the same size (see figure 2), the ESN performs much worse on test sentences. Nevertheless, even at the words where performance is minimal, it is still significantly

[†]If the spectral radius of $\mathbf{W}_{\text{rec}}$ is greater than (or equal to) 1, the network is not considered to be an ESN because it lacks an important property, namely that $\mathbf{a}_{\text{rec}}(t)$ becomes independent of $\mathbf{a}_{\text{rec}}(0)$ as $t \to \infty$ (Jaeger 2003).

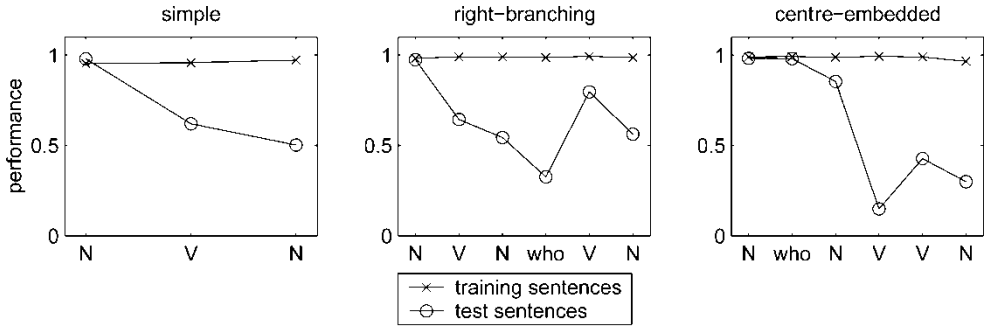

Figure 6. ESN performance at each word of simple, right-branching and centre-embedded training and test sentences ($n = 20$; $w = 18$).

larger than zero, as indicated by sign tests (for simple sentences: $z = 20.7$, $p < 10^{-16}$; right-branching: $z = 11.0$, $p < 10^{-16}$; centre-embedded: $z = 6.36$, $p < 10^{-16}$). That the ESN does not generalize as well as the SRN is also apparent in the percentage of correctly processed complex test sentences, which was only 28.1%.

Comparing an ESN with $n = 20$ to an SRN of the same size is, of course, not fair to the ESN. The SRN's recurrent weights are adapted to the task during training, while the ESN is stuck with its random recurrent network. As is clear from figure 7, manipulating network and lexicon size strongly affects the network's performance. The pattern of performance scores over words, however, does not depend much on the general level of performance, so performance scores can be averaged over words when investigating the effects of n and w.

Figure 8 shows that ESN performance improves as n is increased. Compared to the SRN results in figure 4, ESN behaviour is much more straightforward: bigger is better. This supports the hypothesis that SRN performance on test sentences was limited because of the large number of free parameters in large SRNs. An ESN of the same size runs a smaller risk of overfitting because it has far fewer free parameters, resulting in better generalization.

In figure 9, the effect of varying lexicon size w on generalization by the largest ($n = 100$) ESN is plotted. This effect, too, is more clear-cut than it was for the SRN: larger lexicons lead to better performance on test sentences. The best performing ESN (with $n = 100$ and $w = 42$) is superior to the best performing SRN (with $n = 40$ and $w = 42$). Wilcoxon match-pairs signed ranks tests showed that the difference in test sentence performances is highly significant (for simple sentences: $z = 61.0$, $p < 10^{-16}$; right-branching: $z = 9.87$, $p < 10^{-16}$;

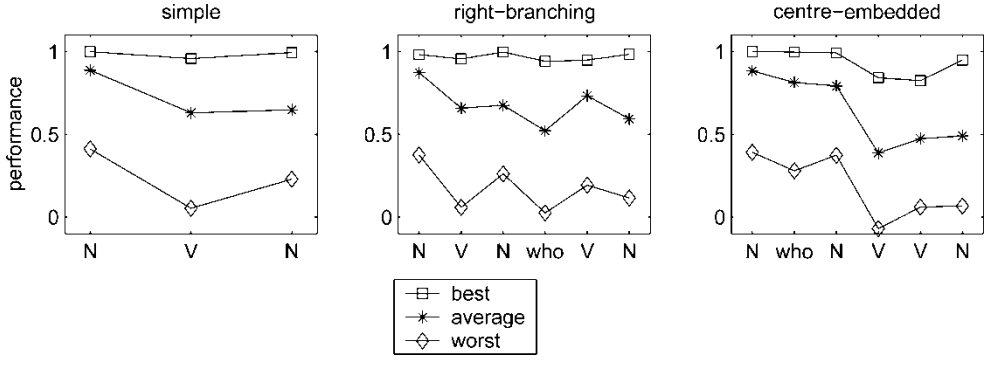

Figure 7. ESN performance, averaged over test sentences, for best and worst levels of n and w, as well as averaged over all levels of n and w.

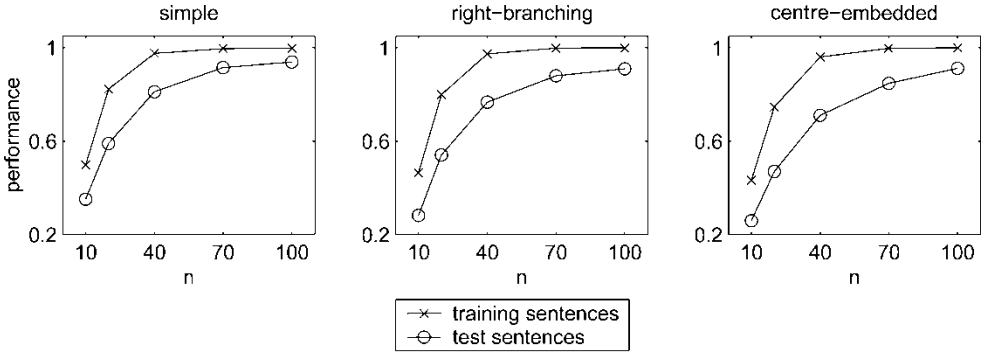

Figure 8. ESN performance as a function of recurrent layer size n, averaged over all words and lexicon sizes of simple, right-branching and centre-embedded training and test sentences.

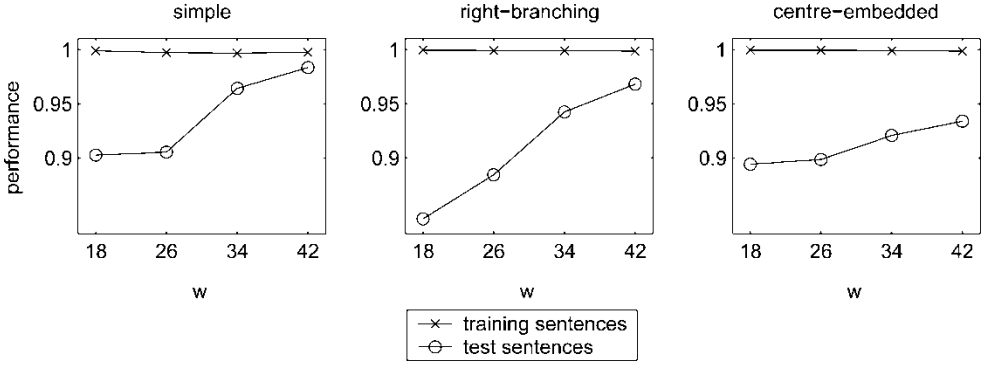

Figure 9. ESN performance as a function of lexicon size w, averaged over all words of simple, right-branching and centre-embedded training and test sentences ($n = 100$).

centre-embedded: $z = 13.0$, $p < 10^{-16}$). The same is true for the percentage of complex test sentences processed correctly, which was 89.0% ($\chi^2 = 19.9$; $df = 1$; $p < 10^{-5}$).

If the number of recurrent units is increased to $n = 1530$, the ESN has the same number of adjustable weights as the SRN with $n = 100$, which suffered from overfitting. However, such a large ESN does not result in worse performance than the much smaller ($n = 100$) ESN. With $n = 1530$ and $w = 42$, average performance scores on simple, right-branching and centre–embedded test sentences were 0.98, 0.97 and 0.96, respectively. For $n = 100$, these figures were 0.98, 0.97 and 0.93. Also, as much as 93.3% of complex test sentences was processed correctly. Even for very large n, the ESN does not seem to overfit.

5. Discussion

5.1 *Recurrent layer size and memory capacity*

Simple recurrent networks display weak systematicity after being trained on only a small fraction of possible sentences, as indicated by their ability to generalize to novel sentences. The simulation results presented here also indicate that varying the number of recurrent units does not improve SRN performance on test sentences to a level comparable with the network's performance on training sentence. Presumably, small networks lack the capacity required for

generalization, while very large networks have so much storage capacity that generalization is not needed for perfectly learning the word-predication task on all training sentences. As a result, test-sentence performance is highest for medium-sized SRNs.

Increasing the number of units in an SRN's recurrent layer can be thought of as increasing the network's capacity in two different ways. First, there is an increase in what might informally be termed 'short-term memory' (STM): the recurrent layer's capacity for storing the input sequence. Second, 'long-term memory' (LTM), the capacity for storing the training examples, increases because there are more connection weights to set. A larger STM is likely to be beneficial in any case, but increasing LTM will result in overfitting when n becomes very large. Note that the number of trainable connection weights (LTM capacity) increases *quadratically* with the number of recurrent units (STM capacity). This makes it more likely that, as n increases, the beneficial effects of increasing STM become overridden by the negative effect of increasing LTM. This may explain the shapes of the performance curves of figure 4.

For ESN, Jaeger (2002) developed a formal account of STM capacity. Given a network that is optimally trained to reproduce input sequences after a delay of k time steps, STM capacity is defined as the fraction of variance in the output accounted for by the input, summed over $k = 1, \ldots, \infty$. Under quite specific conditions (e.g. independently and identically distributed inputs, linear recurrent and output units), the STM capacity of an ESN is proven to equal n, the number of units in its dynamical reservoir. In practice, however, STM capacity is often much smaller than n. Nevertheless, the extent to which information about the input sequence can be extracted from the dynamical reservoir generally increases with n, so increasing the size of the dynamical reservoir can be expected to lead to better results. Indeed, figure 8 shows a clear positive effect of recurrent layer size on the ESN's ability to generalize.

Since an ESN's recurrent weights are not trained, there is a *linear* relation between n and the number of free parameters. This means that, as n is increased, the LTM capacity of an ESN becomes much smaller than that of an SRN with the same recurrent layer size. Consequently, there was no evidence of overfitting by the ESN even when $n = 1530$. Although there is no reason to believe that ESNs will never overfit, this result indicates that the advantage of increasing STM persists for longer in an ESN than in an SRN.

Be reminded from section 4.1 that, in preliminary simulations, a linear dynamical reservoir gave better results than a sigmoidal one. This finding may also be related to STM capacity. Although Jaeger (2002) proved that the maximum STM capacity equals n for both linear and sigmoidal dynamical reservoirs, a linear network was found to result in a larger capacity in practice. In one experimental comparison of a linear and a sigmoidal 400-unit dynamical reservoir (Jaeger 2002: 22–23), the linear network resulted in a capacity of 145, while the sigmoidal one reached only 51 (although it was claimed that an STM capacity close to that of the linear network could be reached by appropriate scaling of input weights).

5.2 *Effect of lexicon size*

As expected, increasing lexicon size was found to have a positive effect on generalization. This effect was much clearer for the ESN than for the SRN. The larger the lexicon, the more important it is for the network to generalize in order to perform its task. Larger lexicons also lead, through combinatorial systematicity, to a larger number of possible sentences and, therefore, to a smaller fraction of sentence types experienced during training (see table 2). This indicates that learning combinatorial systematicity may actually be helped, rather than hindered, by the fact that relatively few sentence types are used for training. Extrapolating the results in figure 9, it seems likely that ESN generalization can be improved further by using even larger lexicons.

5.3 *Conclusion*

The results presented provide evidence that training *less* can lead to learning *more*: reducing the fraction of possible sentences used for training, or the number of trained connection weights, helps the network to learn to generalize and, thereby, to handle weak combinatorial systematicity. Considering the importance of systematicity to language, ESN may therefore make more appropriate models of human sentence processing than SRNs.

Acknowledgements

I would like to thank Michael Klein for his useful remarks on an earlier version of this paper. The research presented here was supported by grant 451-04-043 of the Netherlands Organization for Scientific Research (NWO).

References

K. Aizawa, *The Systematicity Arguments*, Dordrecht: Kluwer Academic, 2003.
E.B. Baum and D. Haussler, "What size net gives valid generalization?", *Neural Comput.*, 1, pp. 151–160, 1989.
M. Bodén, "Generalization by symbolic abstraction in cascaded recurrent networks", *Neurocomputing*, 57, pp. 87–104, 2004.
D.J. Chalmers, "Connectionism and compositionality: why Fodor and Pylyshyn were wrong", *Phil. Psychol.*, 6, pp. 305–319, 1993.
M.H. Christiansen and N. Chater, "Generalization and connectionist language learning," *Mind & Language*, 9, pp. 273–287, 1994.
M.H. Christiansen and N. Chater, "Toward a connectionist model of recursion in human linguistic performance", *Cognitive Sci.*, 23, pp. 157–205, 1999.
J.L. Elman, "Finding structure in time", *Cognitive Sci.*, 14, pp. 179–211, 1990.
J.L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure", *Machine Learning*, 7, pp. 195–225, 1991.
J.L. Elman, "Learning and development in neural networks: the importance of starting small", *Cognition*, 48, pp. 71–99, 1993.
J.L. Elman, "Generalization, simple recurrent networks, and the emergence of structure", in *Proceedings of the 20th Annual Conference of the Cognitive Science Society*, M.A. Gernsbacher and S. Derry, Eds, Mahwah, NJ: Erlbaum, 1998.
J.A. Fodor and Z.W. Pylyshyn, "Connectionism and cognitive architecture: a critical analysis", *Cognition*, 28, pp. 3–71, 1988.
R.F. Hadley, "Systematicity in connectionist language learning", *Mind & Language*, 9, pp. 247–272, 1994.
R.F. Hadley and V.C. Cardei, "Language acquisition from sparse input without error feedback", *Neural Net.*, 12, pp. 217–235, 1999.
R.F. Hadley and M.B. Hayward, "Strong semantic systematicity from Hebbian connectionist learning", *Minds & Machines*, 7, pp. 1–37, 1997.
R.F. Hadley, A. Rotaru-Varga, D.V. Arnold and V.C. Cardei, "Syntactic systematicity arising from semantic predictions in a Hebbian-competetive network", *Connection Sci.*, 13, pp. 73–94, 2001.
H. Jaeger, "Short term memory in echo state networks". Technical Report No. GMD Report 152, German National Research Centre for Information Technology (2002).
H. Jaeger, "Adaptive nonlinear system identification with echo state networks", In *Advances in Neural Information Processing Systems, Vol. 15*, S. Becker, S. Thrun and K. Obermayer, Eds., Cambridge, MA: MIT Press, 2003.
H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication", *Science*, 304, pp. 78–80, 2004.
W. Maass, T. Natschläger and H. Markram, "Real-time computing without stable states: a new framework for neural computation based on perturbations", *Neural Comput.*, 14, pp. 2531–2560, 2002.
M.C. MacDonald and M.H. Christiansen, "Reassessing working memory: comment on Just and Carpenter (1992) and Waters and Caplan (1996)", *Psychol. Rev.*, 109, pp. 35–54, 2002.
K.-R. Müller, M. Finke and N. Murata, "A numerical study on learning curves in stochastic multi-layer feed-forward networks", *Neural Comput.*, 8, pp. 1085–1106, 1996.
D.L.T. Rohde and D.C. Plaut, "Language acquisition in the absence of explicit negative evidence: How important is starting small?", *Cognition*, 72, pp. 67–109, 1999.
P. Tiňo, M. Čerňanský and L. Beňušoková, "Markovian architectural bias of recurrent neural networks", In *Intelligent Technologies: From Theory to Applications*, P. Sincák, V. Kvasnicka, J. Vascák and J. Pospíchal, Eds, Amsterdam: IOS Press, 2002, pp. 17–23.

P. Tiňo, M. Čerňanský and L. Beňušoková, "Markovian architectural bias of recurrent neural networks", *IEEE Trans. Neural Net.*, 15, pp. 6–15, 2004.
F. Van der Velde, G.T. Van der Voort van der Kleij and M. De Kamps, "Lack of combinatorial productivity in language processing with simple recurrent networks", *Connection Sci.*, 16, pp. 21–46, 2004.

Appendix A: Preliminary ESN simulations

A standard ESN does not have a hidden layer between the dynamical reservoir and the output layer. Consequently, ideal output connection weights can be found using a linear regression algorithm. The off-line ESN training method described by Jaeger (2003) was applied to a standard ESN. All units had logistic activation functions. The training set was as described in section 2.1, with lexicon size $w = 18$.

Three parameters were varied: the number of recurrent units (n), the spectral radius of $\mathbf{W}_{\text{rec}}$ ($|\lambda_{\max}|$) and the range of input connection weights (r). Input weights were chosen randomly from a uniform distribution between $\pm r$. As parameter settings, all combinations of the following values were applied: $n = (10, 20, 40, 70, 100, 200, 400)$, $|\lambda_{\max}| = (0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.98, 0.99)$, $r = (0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10)$. The highest performance score (averaged over all 200 complex test sentences) was obtained with parameter setting $n = 100$, $|\lambda_{\max}| = 0.7$, $r = 0.01$. This score was 0.72. In comparison, the SRN of section 3.1 reached a score of 0.92.

The standard ESN does not develop task-specific internal representations of words or sentences. Elman (1991) argued for the importance of such representations to the next-word prediction task. However, simply adding a 10-unit hidden layer between the dynamical reservoir and the output layer does not seem to suffice. Such a network (with softmax output activation function) was trained using backpropagation, as described in section 2.2. Dynamical reservoir size was fixed at $n = 100$, but parameters $|\lambda_{\max}|$ and r were varied as mentioned above. The highest performance score (averaged over all 200 complex test sentences) was only 0.49, with parameter setting $|\lambda_{\max}| = 0.98$ and $r = 1$. As is clear from section 4.2, using linear units in the dynamical reservoir results in much better performance.