

## Lack of combinatorial productivity in language processing with simple recurrent networks

Frank van der Velde\*, Gwendid T. van der Voort van der Kleij\* and Marc de Kamps<sup>†</sup>

\*Cognitive Psychology Unit, University of Leiden, Wassenaarseweg 52, 2333 AK Leiden, The Netherlands

email: {vdvelde, gvdvoort}@fsw.leidenuniv.nl

tel: (31) (0)715273637

fax: (31) (0)715273783

<sup>†</sup>Robotics and Embedded Systems, Department of Informatics, Technische Universität München, Boltzmannstr. 3, D-85748 Garching bei München, Germany

email: kamps@in.tum.de

*Abstract.* An astronomical set of sentences can be produced in natural language by combining relatively simple sentence structures with a human-size lexicon. These sentences are within the range of human language performance. Here, we investigate the ability of simple recurrent networks (SRNs) to handle such combinatorial productivity. We successfully trained SRNs to process sentences formed by combining sentence structures with different groups of words. Then, we tested the networks with test sentences in which words from different training sentences were combined. The networks failed to process these sentences, even though the sentence structures remained the same and all words appeared on the same syntactic positions as in the training sentences. In these combination cases, the networks produced work–word associations, similar to the condition in which words are presented in the context of a random word sequence. The results show that SRNs have serious difficulties in handling the combinatorial productivity that underlies human language performance. We discuss implications of this result for a potential neural architecture of human language processing.

*Keywords:* combinatorial productivity, language performance, word associations, simple recurrent networks.

### 1. Introduction

The study of natural language processing has been dominated by the rule-based approach derived from the development of generative grammar (Chomsky 1957, 2000). In this view, humans are capable of producing and understanding language because they possess knowledge of a grammar that, in combination with a lexicon, allows them to generate (and understand) a virtually unlimited set of sentences. A characteristic aspect of this

approach is the distinction between competence and performance. In particular, the generative capacity of the grammar provides the language user with the competence to produce sentences with unlimited syntactic complexity. However, in actual performance, this unlimited complexity will be restricted due to memory limitations and the like.

In an alternative to this approach, connectionist models have been proposed that relate the limited capacity of performance with the restrictive ability to represent syntactic structures in connectionist networks. In this approach, language processing does not result from the generative use of an (innate) grammar in combination with a lexicon. Instead, language processing results from learning syntactic structures based on the ability to represent (grammatical) sequences of words. To date, the most influential models of this kind are based on so-called simple recurrent networks (SRNs) (Miikkulainen 1996, Christiansen and Chater 1999a, 2001, Palmer-Brown et al. 2002).

An SRN (Elma 1991) consists of a multilayer feedforward network, which can be trained to identify correctly input patterns. In this case, however, the state of the hidden layer of the network is conveyed back to part of the input layer of the network after each training cycle or performance cycle. In this way, the network has information about its prior state when it learns to classify a new input in the training phase or when it classifies a new input in the performance phase. As a result, the network can learn to classify a sequence of input representations.

Elman (1991) used an SRN to predict the next word that would follow at each point in a sentence. For instance, with the sentence *Boys who chase girls feed cats*, the network could reliably predict that after *Boys who chase* a noun would occur in the sentence, and that after *Boys who chase girls* a plural verb would occur. To perform this task, the network was first trained on a set of similar sentences. Each of the sentences used (in training and in the prediction task) was generated on the basis of a phrase structure grammar and a lexicon of about 20 words.

The network could also handle grammatical structures such as centre-embedded relative clauses, but performance of the network began to deteriorate with multiple centre-embeddings. Such a limitation in performance is very interesting because it matches the performance limitation found in human sentence comprehension. Humans have great difficulty in understanding double centre-embedded sentences, to the point that these sentences can be rated as incomprehensible (see, e.g. Gibson 1998).

From the perspective of a generative grammar, syntactic structures such as centre-embedding result from a recursive productivity. The rules of the grammar allow sentences to be embedded in other sentences as relative clauses. By the repeated application of these rules, arbitrary complex sentences can be produced in this way. Christiansen and Chater (1999b) investigated the ability of SRNs to handle recursive structures. They trained SRNs on artificial languages in which different forms of recursive structures occur. In line with the preliminary observation of Elman (1991), they showed that SRNs could be trained to process recursive structures with performance limitations comparable with human language processing.

The fact that performance limitations observed with SRNs match human performance limitations with similar sentence constructions provides a different view on the distinction between competence and performance. In the rule-based approach, sentences with, for instance, double centre-embeddings are grammatically correct and thus belong to the competence of the language user (i.e. the domain of grammatical knowledge of the language user). The performance limitation with these sentences then results from factors outside the language domain, such as limitations in memory or attention span. In contrast, the performance limitations with SRNs are a direct result of the way in which grammatical structure is represented in the network. The network does not

represent a grammar separate from a lexicon. Instead, the network has formed a representation of a statistical average of the set of sentences used to train the network. The performance limitations with recursive structures observed with SRNs reflect the limited way in which these structures can be represented in these networks. In a similar manner, the performance limitations observed with human language processing could reflect the limited representation of recursive structures in natural language. This would entail that SRNs are more adequate models of human language processing than rule-based models (e.g. Elman 1995, Churchland 1995, Christiansen and Chater 2001).

## 2. Combinatorial productivity

The performance limitations in human language processing provide important information about the nature of human language processing. Any model of human language processing should indeed account for these performance limitations (van der Velde 1995). However, to account for human language processing, it is not sufficient to focus on performance limitations alone. Human language is a very powerful means of communication, which would not have come about if language was intrinsically limited in every aspect.

The communicative aspect of language can be described as the ability to provide specific information about ‘who does what to whom’ (e.g. Pinker 1994, Calvin and Bickerton, 2000). Thus, the sentences *John loves Mary* and *Bill loves Susan* provide different information, even though they have the same syntactic structure. The specific information provided by these sentences can be derived from answering specific ‘who does what to whom’ questions, such as ‘Whom does John love?’ and ‘Who loves Susan?’ The ability to answer such questions is a crucial aspect of human language processing.

Human language processing is very productive in the ability to represent ‘who does what to whom’ information. Consider, for instance, the sentence *John loves dogs and Mary loves cats* (or *John walks with dogs and Mary walks with cats* in the case of intransitive verbs). These sentences consist of six ‘open class’ words (e.g. nouns, verbs, proper names) in a particular grammatical structure. Using the same structure, and a ‘lexicon’ of 1000 alternatives for each noun, proper name or verb in this sentence, a set of  $10^{18}$  different sentences can be created. The use of a lexicon of this size is not excessive, given that the average English-speaking 17-year-old knows more than 60 000 words (Bloom 2000). Anyone who understands the grammatical structure of these sentences and who knows the words in the lexicon would be able to answer binding questions of the form ‘who does what to whom’ for any of these sentences, even for sentences that are false or that have no meaning. For instance, with the sentence *cats love dogs and houses eat furniture*, one would be able to answer questions like ‘Who loves dogs?’ or ‘Who eats furniture?’ Yet, the whole set of these sentences is of ‘astronomical’ magnitude (e.g. the estimated lifetime of the universe, expressed in seconds, is of the same magnitude). Moreover, it is clear that this set is only a subset of the set of sentences that can be understood by any language user (i.e. the ‘performance set’ of natural language). Pinker (1998) gives an estimate of  $10^{20}$  for a ‘performance set’ in natural language, which consists of sentences with 20 words or less; but even this set will probably be an underestimate of the true ‘performance set’ of a natural language user.

The issue here is not the recursive productivity of language, which results in sentences of ever increasing complexity. Instead, the issue here is the combinatorial productivity of language. Combinatorial productivity results when even very simple sentence structures are combined with a large but finite lexicon. As a result, and unlike recursive productivity, combinatorial productivity is not unlimited. Yet, as the examples discussed above illustrate, combinatorial productivity results in sets of sentences of astronomical

magnitude, which are nevertheless understandable because they result from combining a finite lexicon with restricted sentence structures.

Combinatorial productivity raises the question of how sets of sentences of this magnitude can be learned with SRNs. To put this in perspective, we shall attempt to make a comparison between the actual learning behaviour of an SRN and the learning behaviour that would be required for natural language as a whole. Thus far, SRNs have been trained and tested on toy grammars with small lexicons (Christiansen and Chater 2001). In the case of Elman (1991), the lexicon contained about ten nouns and ten verbs. The network was trained with a set of sentences in the order of  $10^4$ . The network was tested using sentences with a mean length of about six words. Given the grammar used, these sentences contained at most five nouns and verbs. Thus, the performance of the network was tested using a set of sentences in the order of  $10^5$ . It should be noted that the performance of the network was reasonable but not optimal, which entails that the training set was not too large. Thus, it would seem that, with a training set of about 1–10% of the set on which the performance was tested, the network had a ‘learning-to-performance’ ratio in between 1-to-100 and 1-to-10.

By comparison, the ‘performance set’ of natural language is in the order of  $10^{20}$  sentences or more, as argued above. It is obvious that learning about 1–10% of a set of this magnitude is impossible. Therefore, to deal with a language of the size of natural language a form of combinatorial productivity seems to be needed. This entails that when new words are learned, say in a particular sentence context, these words can also be recognized in other sentence contexts without learning. Thus, when one learns that *Dumbledore is headmaster of Hogwarts*, one can also recognize a sentence like *The dog chases Dumbledore*, even though this sentence was not specifically trained (or the combination of the familiar words *dog*, *chases* and the new word *Dumbledore* did not occur in any of the training sentences). Adult language users possess this form of productivity (e.g. Pinker 1984), but even children seem to possess the ability to use newly acquired words in new sentence contexts (e.g. Katz *et al.* 1974, Ingram 1989). Thus, with combinatorial productivity, new words would only have to be learned in a few sentence contexts. They could then be used in other sentence contexts as well. In this way, the need to train a significant number of sentences of the overall set of sentences can be avoided.

### 2.1. Combinatorial productivity and systematicity

Productivity of language use with SRNs has been a topic of previous research. For instance, Marcus (1998) trained SRNs on two sentence frames. The first one consisted of sentences of the form *the bee sniffs the X*, where *X* could be a word like *rose*, *lily*, *tulip* or *lilac*. The second sentence frame consisted of sentences of the form *a Y is a Y*, in which *Y* could also be a word like *rose*, *lily* or *tulip*. However, the word *lilac* appeared only in the first sentence frame. The network learned each of the sentences, but it could not predict that *lilac* was a plausible continuation of the sentence ‘*a lilac is a . . .*’. Thus, it could not productively use the word *lilac* from the learned sentence *a bee sniffs a lilac* to the novel sentence *a lilac is a lilac*.

The simulation of Marcus (1998) concerned the ability to learn a word in a given sentence frame and then to use it in a novel sentence frame. This form of productivity is referred to as ‘strong systematicity’ (Hadley 1994, Phillips 1998). In general, strong systematicity refers to the ability to use familiar words in novel sentence contexts and/or novel syntactic positions (e.g. as in the *Dumbledore* examples given above). For instance, nouns will be trained in the object position, and then they will be tested in the subject position. Hadley (1994) and Phillips (1998) demonstrated failures of strong systematicity

in connectionist networks, like SRNs. However, one could argue that strong systematicity is too difficult for SRNs, precisely because SRNs have to learn the words and their syntactic use at the same time (to avoid the distinction between lexicon and rules). Furthermore, for combinatorial productivity as defined above, strong systematicity may not be needed. Instead, ‘weak systematicity’ (Hadley 1994) may suffice. In this form of systematicity, each word is trained on all syntactic positions in which it can appear in the (novel) test sentences. For instance, the sentences *boy sees girl* and *dog chases cat* have been learned, and the network is tested with sentences like *dog sees cat* and *boy chases girl*. Thus, the words *sees* and *chases* are transferred to new sentence contexts, but on the same syntactic position on which they have been learned.

In this way, a substantial form of productivity can be achieved. For instance, assume that the lexicon of 1000 nouns and 1000 verbs is partitioned into, say, 100 sets of 10 nouns and 10 verbs each. Each subset of words could then be used to form the sentences of the type *John loves dogs and Mary loves cats*. This results in 100 sets of sentences of this type, and each set consists of  $10^6$  sentences (given by six nouns/verbs for each sentence, with ten alternatives for each noun and each verb). Each set of sentences could then be learned in the manner of Elman (1991), with about  $10^5$  training sentences for each set (10% for each set). As a result, only a set of  $100 \times 10^5$  sentences would have to be used to learn the overall set of  $10^{18}$  sentences, which reduces the set of  $10^{16}$ – $10^{17}$  training sentences to a training set of about  $10^7$ . A set of  $10^7$  training sentences could be feasible for a model of human language processing based on SRNs. In the simulations presented hereafter, we investigated this form of productivity with SRNs.

### 3. Simulations

The type of network used in the simulations is shown in figure 1. The network received an input of one word at a time and its task was to predict the next word that would come, as in Elman (1991). The stimuli in this simulation were based on a lexicon of 18 items. These included eight singular nouns, eight singular verbs, the relative pronoun ‘**who**’ and an end-of-sentence indicator ‘.’. Figure 1 illustrates how the vectors representing the pronoun ‘**who**’, the end-of-sentence indicator ‘.’ and the verbs and nouns are organized in the output layer of the network.

The items were represented as 20-bit binary vectors. Each item was assigned a unique vector in which only one bit was turned on. As a result, all items are represented orthogonal with respect to each other. Table 1 lists the 20-bit binary vectors representing these words.

The lexicon was divided into four separate groups of two nouns and two verbs each. The relative pronoun ‘**who**’ and the end-of-sentence indicator ‘.’ were part of each lexicon group as well. The four lexicon groups are listed in table 2.

Sequences of words formed sentences. For each lexicon group, sentences were formed using only words from that lexicon group (including ‘**who**’ and ‘.’). The training data contained three different types of sentences. The three types are a simple three-word sentence, a sentence with one right branching and a sentence with one centre-embedding. The syntax for each of the three sentence types is given below, together with an example.

**Simple:** Noun Verb Noun .

*Boy sees girl.*

**Right-branching:** Noun Verb Noun **who** Verb Noun .

*Boy sees girl who hears girl.*

**Centre-embedding:** Noun **who** Noun Verb Verb Noun .

*Boy who boy hears sees girl.*

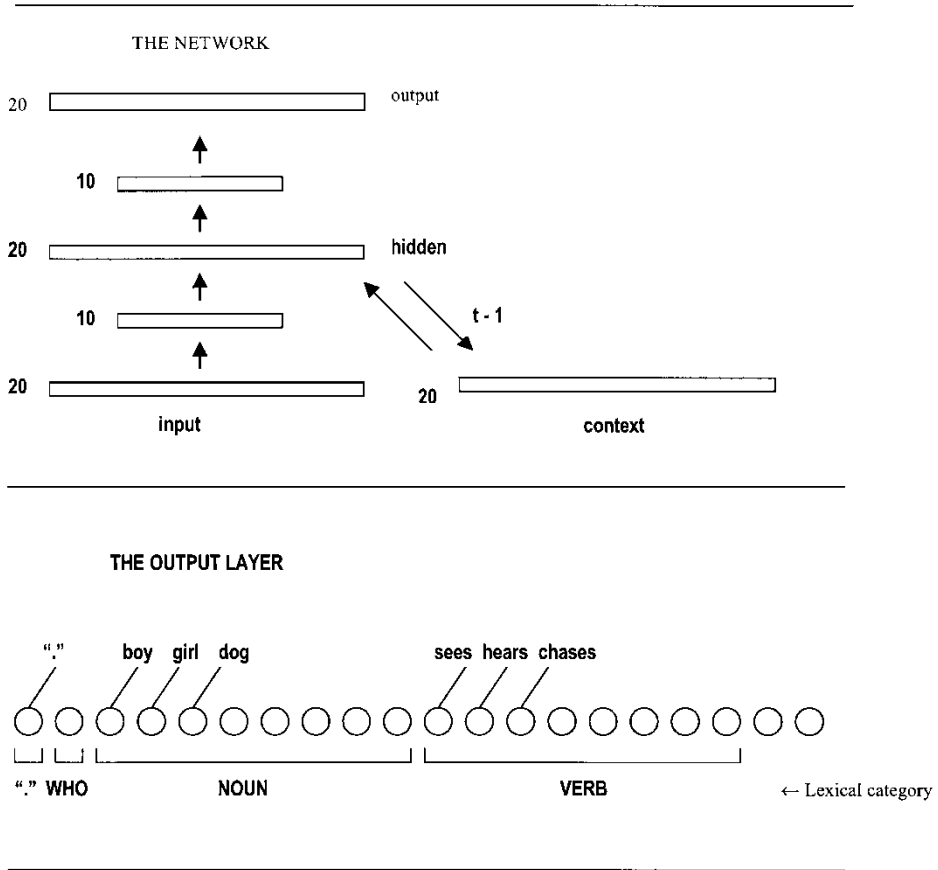


Figure 1. Architecture of the simple recurrent neural networks used in the simulations (top figure). The numbers indicate the number of neurons in a layer. Each circle in the output layer corresponds to one neuron. As an example, for some neurons it is shown which lexical item they refer to.

Since all chosen nouns and verbs were singular and all verbs allowed or required a direct object, all nouns and verbs could fill any of its category places. Thus, all possible combinations of the nouns and verbs could be used to form sentences for each of the three sentence types. The only restriction is that the nouns and the verbs appearing together in one sentence *all* have to be out of the *same* group. The set of training sentences consisted of all sentences formed in this way. Examples of the sentences in each group are presented in the Appendix.

The use of three different sentence types is important. Suppose that the networks were only trained and tested on 'Noun Verb Noun' sentences. In this case, there is a direct correspondence between lexical category (Noun and Verb) and word position (first, second and third) in the training and test sentences. This would allow the networks to learn the association between lexical category and word position, which would eliminate the difference between the training and test sentences.

It is important to note that all sentences of a given sentence type have a common context by which they are related. For the simple (three-word) sentences, a common context for the sentences from the different training groups is provided by the fact that SRNs

Table 1. Coding of the words in vector of 20 bits presented to the input layer of the simple recurrent neural network.

‘.’	10000000000000000000
Who	01000000000000000000
Boy	00100000000000000000
Girl	00010000000000000000
Dog	00001000000000000000
Cat	00000100000000000000
Clarice	00000010000000000000
Anthony	00000001000000000000
Boss	00000000100000000000
Clerk	00000000010000000000
Sees	00000000001000000000
Hears	00000000000100000000
Chases	00000000000010000000
Kicks	00000000000001000000
Follows	00000000000000100000
Loves	00000000000000010000
Obeyes	00000000000000001000
Dislikes	00000000000000000100

have the same kind of representation for dots (‘.’) as for words, and that sentences are presented in concatenated strings to the network (see later). Effectively, this means that all three-word sentences share the same context, that is, they are all of the form ‘. Noun Verb Noun .’. Thus, for instance, the sentences ‘.boy sees girl.’ and ‘.dog chases cat.’ have been learned, and the network is tested with sentences like ‘.dog sees cat.’ and ‘.boy chases girl.’. To increase the common context between the sentences, one could include sentences with combinations like ‘dog sees’, ‘sees cat’, ‘boy chases’ and ‘chases girl’ in the training set. However, we wanted to avoid such combinations in the training set, because the SRNs would then be tested on their ability to reproduce word–word associations instead of sentence structure.

Likewise, all complex sentences of a given type share ‘.’ and ‘who’, which provides a common context for these sentences. Furthermore, the learning of word–word associations in the complex sentences is avoided as well.

### 3.1. Training scheme

In all, ten networks of the type presented in figure 1 were trained and tested with the four lexicon groups presented above. The networks differed in the initial set of random weights before training, so that each network could form a different representation of the set of training sentences. The average performance of ten networks provides a more robust measure of the behaviour of SRNs in this task compared with the performance of a single network.

Each network was trained in four different phases, in line with Elman (1991). Each phase had a different ratio between the number of simple sentences and the number of complex sentences (i.e. right-branched and centre-embedded sentences) in the set of

Table 2. Nouns and verbs appearing in the sentences of each group; the relative pronoun and the end-of-sentence indicator are used in every group.

---

<i>Group 1</i>
N → boy, girl
V → sees, hears
end-of-sentence → ‘.’
relative pronoun → who
<i>Group 2</i>
N → dog, cat
V → chases, kicks
end-of-sentence → ‘.’
relative pronoun → who
<i>Group 3</i>
N → Clarice, Anthony
V → follows, loves
end-of-sentence → ‘.’
relative pronoun → who
<i>Group 4</i>
N → boss, clerk
V → obeys, dislikes
end-of-sentence → ‘.’
relative pronoun → who

---

training sentences. From phase 1 to phase 4, this ratio changed in favour of the complex sentences as follows.

*Phase 1.* The training corpus consisted exclusively of simple sentences. Taking all simple sentences from the four groups, a corpus of 32 sentences with sentence length of four items (each sentence includes the terminal ‘.’) was formed. Each network was trained on 1000 passes through this corpus. A total of 32 000 sentences were thus presented in this phase.

*Phase 2.* The training corpus consisted of all simple sentences and all (256) complex sentences. The ratio between the number of simple sentences and the number of complex sentences was 3 : 1. To ensure a ratio of 3 : 1, the corpus contained a multiple of all simple sentences. As a result, the training corpus consisted of 1024 sentences with a mean sentence length of 4.75. Each network was trained on ten passes through this corpus, giving a total of 10 240 training sentences.

*Phase 3.* The ratio between the number of simple sentences and the number of complex sentences in the training corpus was 1 : 1. The total number of sentences in this corpus was 512 (all complex sentences and a multiple of all simple sentences) with a mean sentence length of 5.5. Each network was trained on 100 passes through this corpus (a total of 51 200 sentences).

*Phase 4.* In this last phase, the ratio between the number of simple sentences and the number of complex sentences in the training corpus was 1 : 4. Mean length of the 320 sentences (twice the number of simple sentences and all complex sentences) in this

corpus was 6.4. Each network was trained on 200 passes through this corpus (a total of 64 000 sentences).

The training sentences in each phase were put in a random order. The sentences were concatenated, so that a network received the first input word of a sentence after the terminal ‘.’ of the previous sentence. During the whole training, each network was presented a sum of 157 440 sentences with a mean sentence length of 5.5 (867 840 items).

### 3.2. Performance

At the conclusion of the fourth phase of training, the weights were frozen at their final values. The network performance was then tested on a set of test sentences, in which nouns and verbs from  $N$  lexicon groups were combined, with  $N$  ranging from 1 to 3 for simple sentences (due to the sentence length of this type) and with  $N$  ranging from 1 to 4 for right-branching and centre-embedded sentences. The sentences with  $N=1$  were sentences from the training set. In the sentences with  $N=2$ , the words were chosen alternatively from two groups, thus the first word from one group, the second word from another group and the third word again from the first group (etc.). The sentences with  $N=3$  and  $N=4$  combined three and four lexicon groups in this way. For each sentence type and for every  $N$  all possible sentences of this kind were formed, from which 32 sentences were randomly selected. These were concatenated as before and used as the testing set for the network.

It is important to notice that the sets of sentences with  $N=2$ ,  $N=3$  and  $N=4$  provide three different tests of combinatorial productivity, with increasing difficulty. Thus, it is possible that the networks could handle sentences with  $N=2$  but would fail on sentences with  $N=3$  and (or)  $N=4$ . In the case of  $N=2$ , the network is trained with sentences like *dog chases cat* and *boy sees girl*, and it is then tested with sentences like *dog sees cat* and *boy chases girl*. Thus, only one word (e.g. *sees*) is inserted into a trained sentence like *dog chases cat*. Furthermore, the inserted word itself is trained in an  $N$  *sees*  $N$  context. In the case of a trained sentence like *dog chases dog who kicks cat*, the network is tested on a sentence like *dog hears dog who sees cat*. In this case, the inserted words *hears* and *sees* are trained in a context like  $N$  *hears*  $N$  *who sees*  $N$ . Examples for  $N=3$  and  $N=4$  are given in the Appendix.

The networks were tested on the ability to predict the lexical category of the word that would follow after a particular sentence context (the terminal symbol ‘.’ and ‘**who**’ are lexical categories in this respect). Thus, in the case of the sentence *dog hears dog who sees cat* the networks were tested on the ability to predict that, for instance, a noun would follow after *dog hears*. Notice that with certain sentence contexts, more than one lexical category can be correct (for the reason explained above). For instance, if the sentence context is ‘Noun Verb Noun’, the next lexical category could be the terminal symbol ‘.’ (forming a simple sentence) or it could be the word ‘**who**’ (forming a right-branching sentence). This entails that, for each input word presented to the network, the set of output units can be divided into a subset of lexically correct units and a subset of lexically incorrect units. To produce a correct prediction of the lexical category of the next word following a given input word, the network should activate one or more of the lexically correct output units. In contrast, any activation of the lexically incorrect output units represents an error made by the network in predicting the next lexical word category. Thus, the sum of activation of the lexically correct units can be labelled as the correct activation produced by the network, and the sum of activation of the lexically incorrect units can be labelled as the incorrect activation produced by the network.

In this way, the grammatical prediction error (*GPE*) was obtained as follows:

$$GPE = \frac{\text{incorrect activation}}{\text{correct activation} + \text{incorrect activation}}.$$

The *GPE* gives a measure of the inability of the networks to predict the next lexical category, given a particular sentence context. *GPE* was based on the average of all sentences of the same type and all ten simulated networks.

Next to this *GPE*, a network's performance was also evaluated according to a winner-takes-all principle. In this way, a network's highest output is taken to be its prediction. The lexical category of this unit is compared with the given sentence context to see whether this prediction is grammatical or ungrammatical. The percentage of grammatically incorrect predictions, averaged over all sentence types and all networks, gives the grammatical prediction error according to this winner-takes-all principle, named *GPE<sub>Wta</sub>* here:

$$GPE_{Wta} = \frac{\sum \text{ungrammatical predictions}}{\sum \text{predictions}}.$$

### 3.3. Results

Figure 2 shows the results of the ten trained networks on the test data, with the performance measured by the *GPE*. Performance is shown as a function of the number ( $N$ ) of lexicon groups combined into one sentence, and of the position of the input word in a sentence. The figure shows the results for the simple sentences (top panel), the right-branching sentences (middle panel) and the centre-embedded sentences (bottom panel). Notice that the graphs for  $N=1$  represent the results on the trained sentences (averaged over all four lexicon groups). Figure 3 shows the results of the ten trained networks on the test data, with the performance measured by the *GPE<sub>Wta</sub>*. The layout is the same as in figure 2. The results depicted in figures 2 and 3 are very similar for each sentence type. Apparently, evaluating the networks by taking only the highest predicted lexical item into account reflects their total activation behaviour. It is clear that the networks performed very well on the set of trained sentences, as reflected in the low *GPE* and *GPE<sub>Wta</sub>* for  $N=1$  with all sentence types.

The mean *GPE* of the predictions by the networks for the  $N=1$  sentences was 0.061. This low value reflects the low percentage of network activation that corresponded to ungrammatical predictions with the trained sentences. The mean *GPE* and mean *GPE<sub>Wta</sub>* were much higher for the  $N>1$  sentences, with 0.250 for  $N=2$ , 0.273 for  $N=3$ , and 0.309 for  $N=4$ . Thus, the networks made substantially more ungrammatical predictions on the sentences that combined words from more than one lexicon group. An analysis of variance confirmed that the differences in *GPE* and in *GPE<sub>Wta</sub>* for the different values of  $N$  are significant ( $F=61.376$ ,  $p<0.001$  and  $F=65.471$ ,  $p=0.001$ , respectively). *Post hoc* Tukey tests revealed that these results should be attributed mainly to the differences in performance between the  $N=1$  condition and  $N>1$  conditions.

The results show that the use of combinatorial productivity has failed. The networks perform very well on sentences that consist of words from one lexicon group (the trained sentences); but when words from different groups are combined, the performance of the networks deteriorates strongly, even though the grammatical structure of the sentences has not changed, and even though all words appear in the same syntactic positions as in the training sentences.

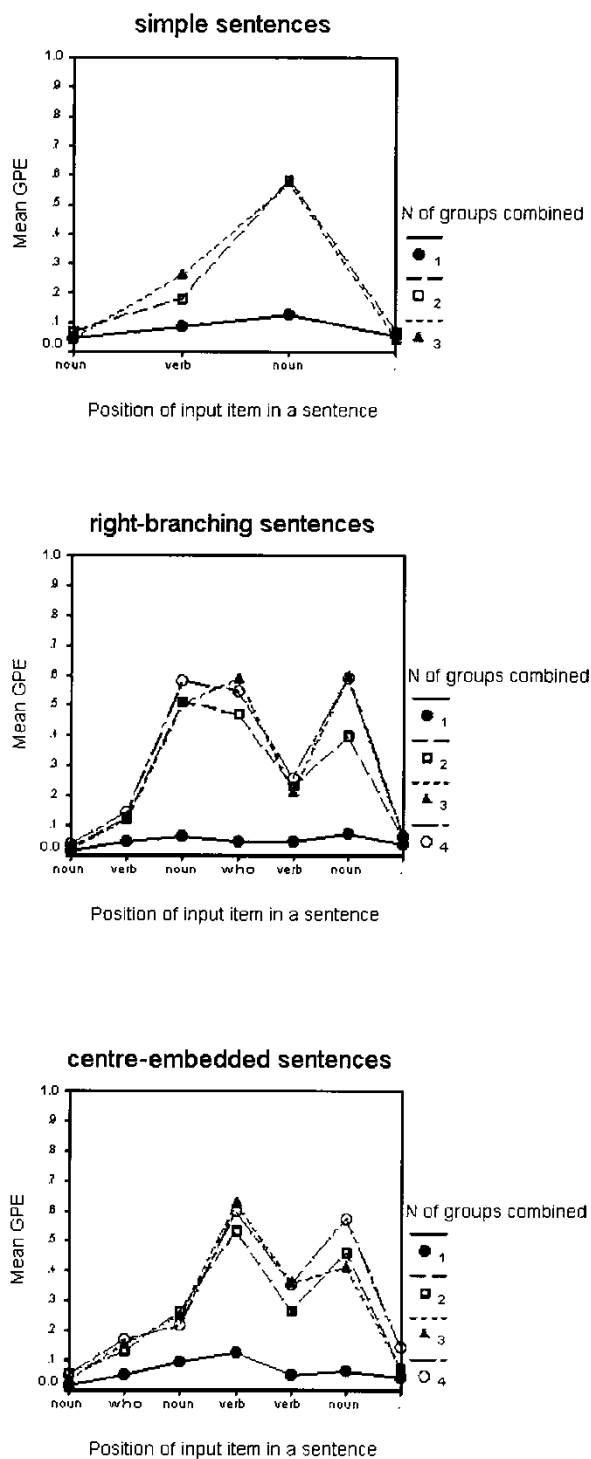


Figure 2. Grammatical prediction error (*GPE*) of networks as a function of  $N$  and of the sentence position of an input word, on simple sentences (top panel), right-branching sentences (middle panel) and centre-embedded sentences (bottom panel).

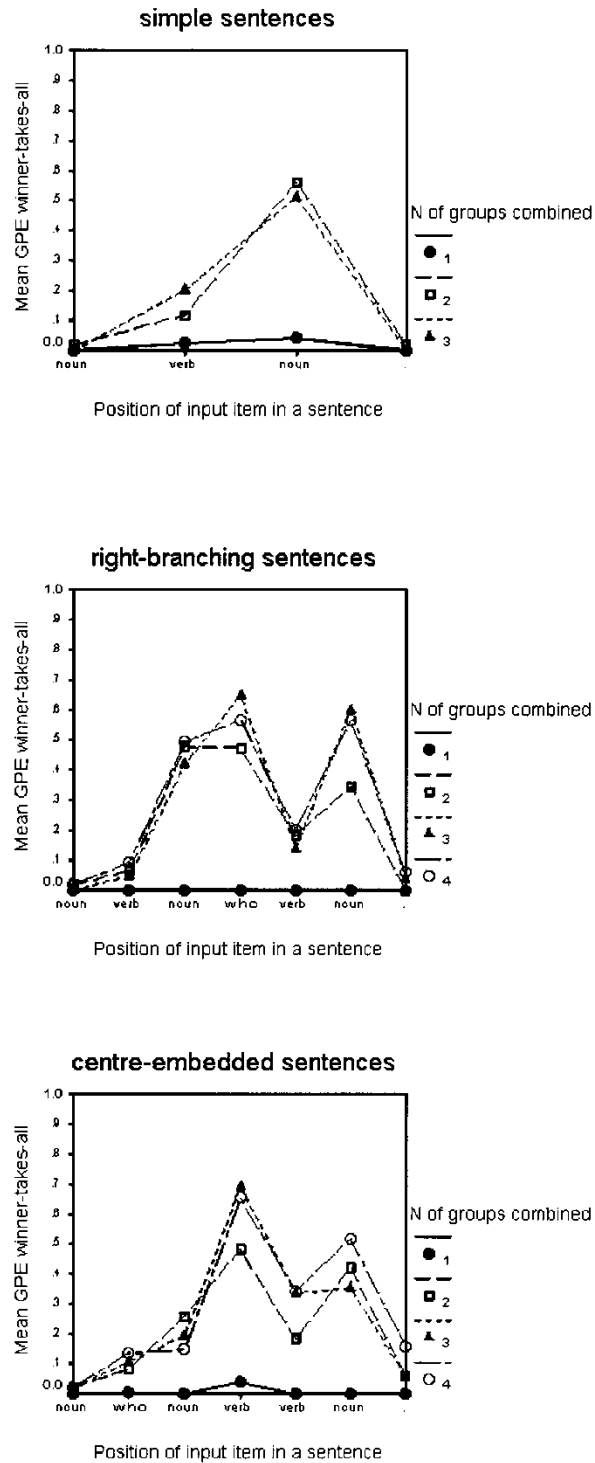


Figure 3. Grammatical prediction error winner-takes-all ( $GPE_{wta}$ ) of networks as a function of  $N$  and of the sentence position of an input word, on simple sentences (top panel), right-branching sentences (middle panel) and centre-embedded sentences (bottom panel).

### 3.4. Trend

A distinctive feature of the simulation results presented in figures 2 and 3 is the similar trend for the  $N > 1$  data within each of the three sentence types (simple, right-branching and centre-embedded sentences). We shall discuss the trend for each of the three sentence types in turn (given the similarity between the *GPE* and the *GPE<sub>wt</sub>* results, we shall consider only the *GPE* results from now on).

With the simple sentences, the difference between  $N = 1$  and  $N > 1$  begins with the second word. This results from the fact that each noun has been used as the first noun of one of the trained sentences. Thus, the networks will treat the first noun as the first word of one of the trained sentences, which results in the same *GPE* for  $N = 1$  and  $N > 1$ . With  $N > 1$ , the second word will be from a different lexicon group than the first word, which results in an increase of *GPE* at that position. When only two words are combined, the conditions  $N = 2$  and  $N = 3$  are the same. A difference between  $N = 2$  and  $N = 3$  can only occur when three words are presented, but the *GPE* is the same for these two conditions. The *GPE* for  $N > 1$  is higher for the third word position compared with the second word position, which indicates that the networks increasingly lose track of the sentence structure when three words are presented in the  $N > 1$  conditions. In the case of  $N = 2$ , a three-word sentence results from taking the first and the third word from one group, and the second word from another group. As illustrated in figure 2, this gives a similar performance as with  $N = 3$ , in which case all three words come from different lexicon groups. Finally, when the terminator symbol ‘.’ is presented, the difference between the  $N > 1$  and the  $N = 1$  conditions disappears.

With the right-branching sentences, the trend in the data is again similar for the  $N > 1$  conditions. As with the simple sentences (and for the same reason), the difference between  $N > 1$  and  $N = 1$  begins with the second input word. The first three words are of the form ‘Noun Verb Noun’, as with the simple sentences. Thus, the *GPE* pattern is the same in both cases (the network has no way of knowing whether it is dealing with a simple sentence or with a right-branching sentence at this point). The fourth word is ‘**who**’ which gives about the same *GPE* as the third word. Notice that each ‘Noun **who**’ combination belongs to one of the trained sentences, which might explain the lack of increase in *GPE* at this point (nevertheless, *GPE* is still substantial at this point). However, performance improves for the verb position after ‘**who**’, giving a substantially lower *GPE* at this point. Again, each ‘**who Verb**’ combination belongs to one of the training sentences, but the decline in *GPE* is nevertheless remarkable. One would expect that the *GPE* would increase or at least stay on the same level if the networks lose track of the sentence structure. Yet, after this improvement, the *GPE* increases again, until the terminal symbol is presented.

A similar pattern can be observed with the centre-embedded sentences. Here, a difference in *GPE* can only be observed after the second input word, because the first two input words consist of the combination ‘Noun **who**’, which belongs to the training set for all nouns. After the second input word, *GPE* increases with each word position, until the position of the second verb is reached. The decline in *GPE* at this point is perhaps even more remarkable because the combination ‘Verb Verb’ does not occur in the training sentences (that is, with the words in the  $N > 1$  conditions). After this decline, however, *GPE* increases again until the terminal symbol is reached.

### 3.5. Word–word associations

As stated above, one would expect that the networks would lose their ability to predict the structure of a sentence when sentence length increases in the  $N > 1$  conditions, because more words have been mixed when sentence length increases. As a result, the

*GPE* would increase or at least stay the same in this situation. Instead, however, a decline in *GPE* occurs for one of the sentence positions in the complex sentences. Furthermore, this decline occurs for each  $N > 1$  condition, even though one might expect that the networks will be more confused when more words from different lexicon groups are combined.

The explanation of this trend may be found in the result for the terminal symbol ‘.’. The terminal symbol produces a very low *GPE* for each sentence type and for all  $N$  conditions. The prediction to be generated after the terminal symbol is the lexical category ‘Noun’; but the combination ‘. Noun’ is found in the training sentences for all nouns in each of the lexicon groups. In fact, the combination ‘Noun . Noun’ occurs in the training sentences for all nouns. Thus, the networks could have learned the association between ‘.’ and each of the nouns. The prediction ‘Noun’ after ‘.’ could result from these learned associations, which would result in a low *GPE* regardless of the sentence context.

In a similar manner, the networks could predict all lexical categories on the basis of such word–word associations for the  $N > 1$  conditions. To test this prediction, we estimated the *GPE* on the basis of word–word associations for each word position in each of the three sentence types presented in figure 2. The calculation proceeded as follows. First, we looked at the transitions between lexical categories that occur in the set of training sentences. Then, we looked at the frequency with which each transition occurs in the training set, in particular in the set of complex sentences. (The word ‘**who**’ does not occur in the simple sentences, and the fourth training corpus is dominated by the complex sentences.)

For instance, with ‘Noun’ as the input word, the lexical transitions that occur in the training sentences are ‘Noun Verb’, ‘Noun **who**’ and ‘Noun.’. Furthermore, in each lexicon group, each input vector representing ‘Noun’ can be associated with two output vectors representing ‘Verb’, one output vector representing ‘**who**’ and one output vector representing ‘.’. Given that the lexical transitions ‘Noun Verb’, ‘Noun **who**’ and ‘Noun.’ occur equally often in the complex sentences, one can assume that an input vector representing ‘Noun’ is associated equally with four output vectors during the training of the networks.

On the basis of this assumption, an input vector representing ‘Noun’ will produce a *GPE* of 0.5 when ‘Verb’ has to be predicted. When ‘**who**’ or ‘.’ has to be predicted it will produce a *GPE* of 0.75. The other associations can be calculated in a similar manner. Because the transition ‘Verb . Noun’ occurs three times more often than the transition ‘Verb . Verb’, the *GPE* for the prediction ‘Noun’ after ‘Verb’ is taken as 0.25, and the *GPE* for the prediction ‘Verb’ after ‘Verb’ is taken as 0.75.

The *GPE* results for the word–word associations based on this assessment are presented in figure 4, together with the *GPE* results for the  $N = 1$  and the  $N > 1$  conditions (figure 2). The results are given for the second input word onwards in case of the simple sentences and the right-branching sentences, and for the third input word onwards for the centre-embedded sentences (for the reasons explained above).

Figure 4 shows that the trend observed in figure 2 can be explained in terms of word–word associations. Thus, when words from different lexicon groups are mixed, the networks do not predict the next word (lexical category) on the basis of the sentence context, but primarily on the basis of the associations between the words learned during training.

An exception is found with the third input word in the centre-embedded sentences (the first input word presented in figure 4). Here, the *GPE* produced on the basis of word–word associations is clearly higher than the *GPE* produced by the network for all  $N > 1$  conditions (which in turn is higher than the *GPE* produced in the  $N = 1$  condition).

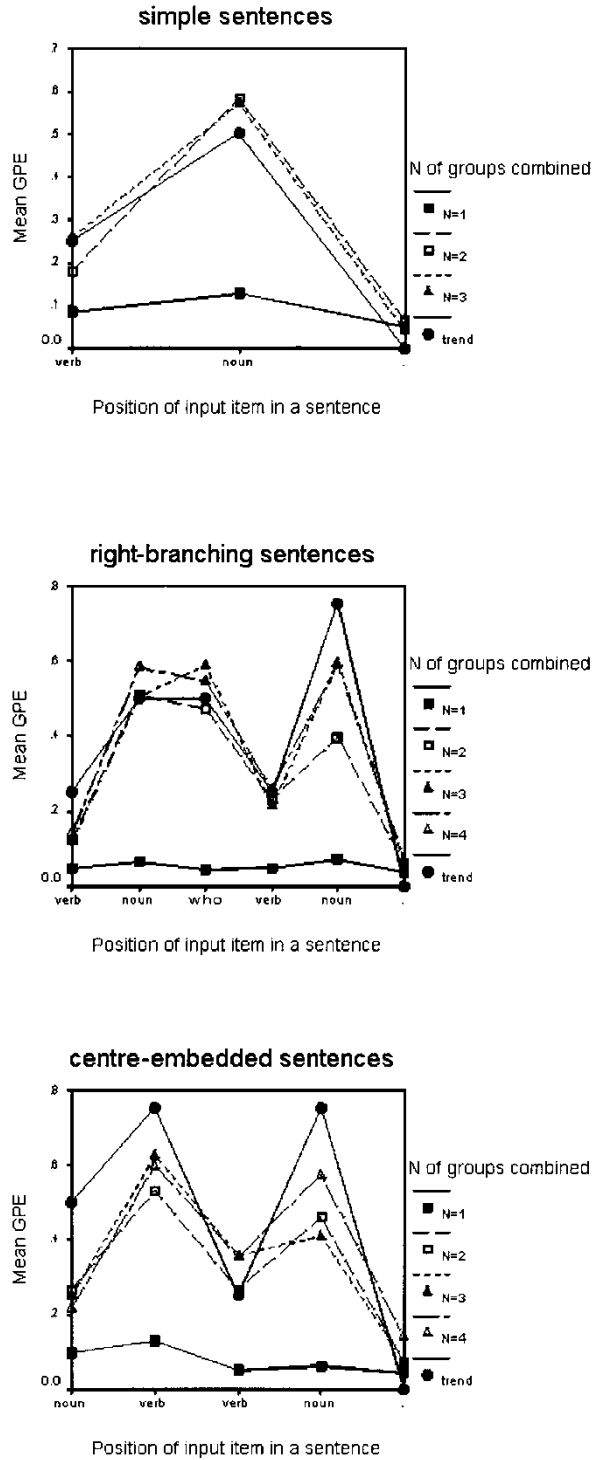


Figure 4. Grammatical prediction error (*GPE*) for the  $N=1$  and the  $N>1$  conditions, and for the word–word associations (the ‘trend’ line), for simple sentences (top panel), right-branching sentences (middle panel) and centre-embedding (bottom panel).

However, the combination ‘Noun **who**’ (the first two words in the centre-embedded sentences) occurs in all trained sentences. The combination of these two words could result in a partial prediction of the category ‘Verb’ for the input word on the fourth position, thereby improving the result for the prediction generated with the third input word. A similar effect was observed by Christiansen and Chater (1994). They found that when a new word like ‘boy’ was used in the context ‘John and boy’, the prediction for the next verb (e.g. ‘chase’) would still be plural, even though the combination ‘John and boy’ was never used in the training set. It seems that the combination ‘John and’ can produce the prediction that the word after the next word should be a plural verb, in line with the prediction after the third input word in the centre-embedded sentences presented in figure 4. However, as illustrated in figure 4, when more words from different lexicon groups are mixed, the *GPE* for the centre-embedded sentences increases and approaches the trend given by the word–word associations.

### 3.6. Testing word–word associations

Instead of calculating the trend for word–word associations based on the assumptions as described above, we also produced the trend for word–word association on the basis of network simulations. If word–word associations dominate the response of the networks in the  $N > 1$  conditions, they will do so in particular when sentence context is not available. This will occur in a ‘word-salad’ condition, in which the context of the test word consists of a random sequence of words. We used a word-salad of seven words (equal to the length of the complex sentences), which were chosen at random from the set of words presented in table 1. After the word-salad was presented, the test word was presented as input and the activations produced by the test word were used to calculate the *GPE* for each of the lexical transitions that could occur in the set of training sentences. Thus, if the test word was a noun, the *GPE* was calculated for the transitions ‘Noun Verb’, ‘Noun **who**’ and ‘Noun .’. Each word presented in table 1 was used as a test word, using 100 different word-salads as sentence context. A few examples are given in the Appendix.

The *GPE* result for the word-salad condition is presented in figure 5, together with the average *GPE* for the  $N > 1$  conditions and the *GPE* for the  $N = 1$  condition. Again, the results are presented for the second input word onwards in the case of the simple sentences and the right-branching sentences and for the third input word onwards in the case of the centre-embedded sentences.

Overall, it is clear that the trend produced by the word-salad condition is similar to the trend produced by the  $N > 1$  conditions. Thus, the  $N > 1$  conditions behave as if the sentence context consists of a random sequence of words (word-salad). In particular for the second input word in the simple and right-branching sentences, and for the third input word in the centre-embedded sentences, the *GPE* for the  $N > 1$  conditions is lower than the *GPE* for the word-salad condition. As discussed earlier, this could result from word associations between words on the first and third (or second and fourth) position in a sentence. The words in the  $N > 1$  conditions are always presented in word positions in which they appear in the set of training sentences. As a result, associations can be formed not just between adjacent words in the learned sentences but also between words that are separated by one word position. These associations could have resulted in the slightly better *GPE* results for the average of the  $N > 1$  conditions over the word-salad condition in the case of the complex sentences.

### 3.7. Combinatorial productivity with mixed word relations in the training set

In constructing our set of training and test sentences we argued that all sentences of a given type are embedded in the same context, such as ‘. Noun Verb Noun .’ in the

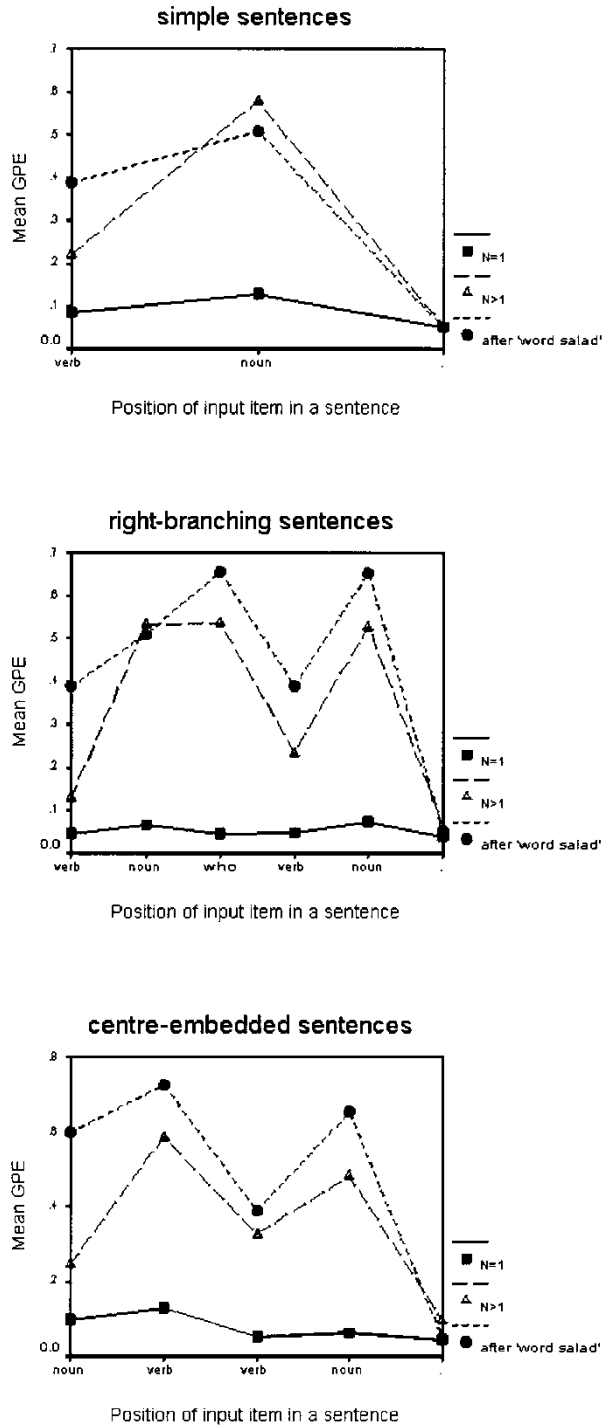


Figure 5. Grammatical prediction error (*GPE*) for the  $N=1$  condition, the average of the  $N > 1$  conditions, and for the 'word-salad' condition, for simple sentences (top panel), right-branching sentences (middle panel) and centre-embedded sentences (bottom panel)

case of three-word sentences. Here, we used the fact that in SRNs a ‘.’ has the same kind of representation as a word. Thus, the ‘.’ at the beginning and the ‘.’ at the end provide the same context for each three-word sentence in each of the four groups.

Nevertheless, one could argue that the words from different groups are not related in terms of grammatical relations in the set of training sentences. As a result, the SRNs would not learn to relate these words in this way, and thus would fail when the words are combined in the test sentences. To test this assumption, we created a new set of training sentences (see Appendix) in which words from the different groups were mixed, so that grammatical relations between these words could be learned.

In the case of three-word sentences we again used test sentences like *boy hears girl* and *dog sees cat* and training sentences like *boy sees girl* and *dog hears cat*. As before, we wanted to avoid combinations like *boy hears* and *hears girl* in the training set. The results presented thus far show that SRNs are indeed very good at learning these word-word associations, thus they will strongly influence the results on the test sentences. Therefore, to introduce (more) grammatical relations between the words in the test and training sentences, we now included training sentences like *boy follows cat* and *dog loves girl*.

To understand the effect, consider the test sentence *boy hears girl*. In this test sentence, we introduce *girl* as the object of *hears*; but in the training sentences we already have *girl* as the object of a verb of which *dog* is a subject (*dog loves girl*), and we have *dog* as the subject of *hears* (*dog hears cat*). Furthermore, we have *girl* as the object of a verb of which *boy* is a subject (in *boy sees girl*).

Likewise, we introduce *boy* as the subject of *hears* in this test sentence; but in the training sentences we already have *boy* as the subject of a verb of which *cat* is the object (*boy follows cat*), and we have *cat* as the object of *hears* (*dog hears cat*). Furthermore, we have *boy* as the subject of a verb of which *girl* is the object (*boy sees girl*).

In this way, we have introduced a chain of relations in the training sentences between the verb *hears* and the arguments *boy* and *girl* in the test sentence *boy hears girl* (short of the direct combinations *boy hears* and *hears girl*).

We used the same procedure with the right-branching sentences. For instance, we used test sentences like *boy kicks girl who dislikes clerk* and *dog chases cat who obeys boss*, and training sentences like *boy chases girl who dislikes clerk*, *dog kicks cat who obeys boss*, *boy obeys cat who loves Anthony* and *dog dislikes girl who follows Clarice*. With these sentences, a similar chain of relations is introduced as with the three-word sentences. Furthermore, we interchanged only one word with these sentences. In this way, the overall context between the test and training sentences is at a maximum (e.g. *boy V girl who dislikes clerk* occurs in both the test and training sentences).

We used the centre-embedded sentences to strengthen further the relations between the words in the test and training sentences for the cases presented above. Thus, we trained sentences like *boy who cat hears obeys Clarice* and *girl who dog hears obeys clerk*. These sentences directly introduce relations between the words *boy*, *hears* and *girl*, (but not in the combinations *boy hears* and *hears girl*). Moreover, the sentence *girl who dog hears obeys clerk* in fact introduces *girl* as the object of *hears*.

The results of the simulations are presented in figure 6. The figure shows that, again, all training sentences were learned to near perfection. In contrast, the SRNs again failed on the test sentences, both with the three-word sentences and with the right-branching sentences. In the case of the right-branching sentences, only one word was changed in the test sentences. From the word ‘**who**’ onwards, the test sentences were the same as the training sentences. The behaviour of the SRNs from that word onwards is also similar for the test and training sentences. This emphasizes the importance of word-word

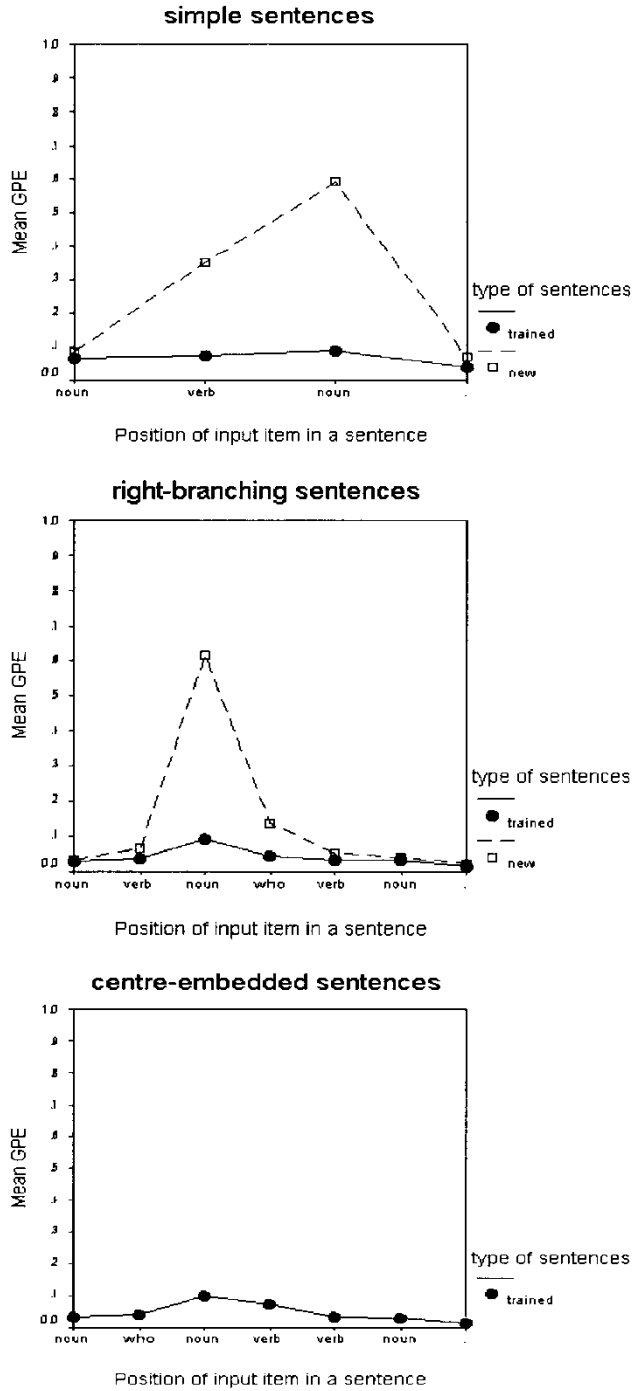


Figure 6. Grammatical prediction error (*GPE*) for the mixed word relations in the training and test sentences for simple sentences (top panel) and right-branching sentences (middle panel) and for the mixed word relations in the training sentences for centre-embedded sentences (bottom panel).

associations for the behaviour of SRNs, because it shows that the disruption before the word ‘who’ in the test sentences has no effect once this word is presented to the networks.

#### 4. Discussion

The networks we simulated failed on the test of combinatorial productivity. The networks performed very well on the set of training sentences. Thus, the networks were capable of simultaneously learning sentences of three different grammatical sentence types; but the networks failed when words from sentences of the same type were combined, even though all words in these sentences appeared on the same syntactic positions as in the training sentences. For instance, in the final simulations (figure 6), the SRNs were able to process sentences like *boy sees girl*, *dog hears cat*, *boy follows cat*, *dog loves girl* and even *girl who dog hears obeys clerk*, yet they failed with a sentence like *boy hears girl*. This behaviour illustrates a failure on a very minimal form of combinatorial productivity.

To assess the relevance of the simulations we presented, it is important to eliminate the potential misunderstanding that we have just presented a negative result, which could have been different if we had used a better learning (training) algorithm. This view is not correct on two accounts. First of all, the learning algorithm we used worked perfectly. All trained sentences were learned to near perfection. The best that another algorithm could do was to learn these sentences to the same level of perfection. It is difficult to see how this could have produced a different result on the test sentences. But even if it did, it would be irrelevant. The crucial point is that our results are not about learning at all, they are about behaviour. We have produced a kind of behaviour with SRNs that you do not find with humans. The SRNs behave in a way that they understand *boy sees girl* and *dog hears cat*, and still fail on *boy hears girl*. This behaviour is not found in humans, regardless of the learning procedure used. It is not found in humans, because the structure of the language system prohibits it. This is what the issue of systematicity is all about: if you understand *boy sees girl* and *dog hears cat*, you cannot but understand *boy hears girl*. Any failure to do so would be regarded as pathological. In fact, the failure of the SRNs in the final simulations (figure 6) is not even comparable to pathological human behaviour. Broca’s aphasics, for instance, often fail on sentences like *girl who dog hears obeys clerk* but they can still understand sentences like *boy hears girl* (Grodzinsky 2000). To the best of our knowledge, the reverse behaviour, as demonstrated by the SRNs, has not been found in human pathological behaviour.

In other words, we have shown a contrast in behaviour with SRNs that cannot be found in a productive language system. This is not a negative result at all. Instead, it is a clear falsification of the assumption that SRNs can handle human language productivity.

In our paper we have focused on the issue of combinatorial productivity because of its relevance for natural language processing (NLP). To understand why, it is important to distinguish between the issues of combinatorial productivity and recursive productivity. Combinatorial productivity concerns the ability to handle a very large lexicon, even in the case of simple and limited syntactical structures. Recursive productivity, on the other hand, deals with the issue of processing more and more complex syntactic structures, such as (deeper) centre-embeddings. We shall illustrate this difference with the long short-term memory recurrent neural networks (LSTMs).

LSTMs outperform standard recurrent neural networks (including SRNs) on the issue of recursive productivity (Gers and Schmidhuber 2001). SRNs are limited in terms of recursive productivity (as humans are!), but LSTMs are not. For instance, they can handle a context-free language like  $a^n b^m B^m A^n$  for arbitrary  $n$  and  $m$  (Gers and

Schmidhuber 2001). Note, however, that this is most likely beyond the capacity of humans (so, we are again faced with the performance versus competence issue).

Furthermore, because of the way in which LSTMs process such languages, they cannot handle any form of combinatorial productivity (unlike humans, who are very good at that). An LSTM is basically an SRN in which a hidden unit is replaced with a ‘memory block’ of units. During learning, the nodes in this memory block develop into counters. In the case of the language  $a^n b^m B^m A^n$ , the network develops two counters, one that counts the number of  $ns$  and one that counts the number of  $ms$ . Thus, one counter counts whether  $a^n$  matches  $A^n$ , and the other whether  $b^m$  matches  $B^m$ . In the context of this language (and the other languages trained with LSTMs), this makes sense because all sentences of this language have the same words, that is, they are all of the form  $a^n b^m B^m A^n$ . The only aspect in which sentences differ is in the value of  $n$  and/or  $m$ , so the system can learn from previous examples that it has to count the  $ns$  and  $ms$ . But this hardly, if at all, makes any sense in the case of NLP. The most fundamental aspect of sentences in natural language is that they convey a message, not that they differ on a given variable (e.g. the number of nouns or verbs). So, the sentence *mouse chases cat* is fundamentally different from the sentence *cat chases mouse*, even though they are both  $NVN$  sentences. How could an LSTM capture this difference? The counters it develops are useless here. What should be counted: the number of times *mouse* and *cat* appear in a sentence? Just consider the number of possibilities that would have to be dealt with, given a lexicon of 60 000 words (the average lexicon of a 17-year-old), not just four words as in  $a^n b^m B^m A^n$ . Furthermore, how would such a model ever deal with novel sentences, like *Dumbledore chases mouse*? How could it have developed counters for the match between *Dumbledore* and *mouse* if it has never seen these words in one sentence before?

Networks such as LSTMs clearly demonstrate why the topic of combinatorial productivity is very important. The issue of combinatorial productivity is essential in NLP and virtually non-existent in artificial languages. Thus, the fact that there are more powerful systems like LSTMs that are capable of processing more complex artificial languages than SRNs proves nothing about their ability to deal with combinatorial productivity. Indeed, as the analysis given above shows, they could be further removed from that ability than ever.

In our simulations, we used word representations that were totally uncorrelated (see table 1). One could argue that this is not in line with the way in which words are represented in the brain. Words could be represented by means of cell assemblies (Pulvermüller 1999), distributed over the brain. Words of a similar category, like verbs or nouns, would have a common part in their cell assemblies that reflects that they are verbs or nouns. This would introduce a common representation for verbs or nouns to which the SRNs could respond. We agree with this view about word representation in the brain, but we did not use it for a specific reason.

To see why, it is important to understand that the work on SRNs is a reaction to the classical view about NLP. In this classical view, NLP consists of a rule system that operates on a database in the form of a lexicon. The rules themselves are represented in abstract terms (e.g. about  $NVN$  sequences). The subsymbolic approach, embodied with SRNs, aims to eliminate this distinction. In the words of Miikkulainen (1996; p 47), the subsymbolic approach to sentence parsing offers the promise that: ‘It is possible to combine syntactic, semantic, and thematic constraints in the interpretation’. This is why, in the case of NLP, SRNs operate on word-strings directly, not on abstract strings such as  $NVN$  strings.

The point about cell assemblies and correlated word representations in general is that they basically reintroduce the distinction that SRNs wanted to avoid. If the cell

assemblies were used as input, the SRNs would learn to react to this common part, that is, they would process a sentence in terms of a sequence of *Ns* and *Vs*. But this would reintroduce the distinction between rule-based operations and the lexicon. In particular, it would eliminate all hope of combining syntactic and semantic aspects of sentence interpretation in one approach.

Nevertheless, it is interesting to investigate this option a bit further. The lack of combinatorial productivity observed here does raise the question of whether SRNs can process a language of the size of natural language by directly processing word strings. However, the SRNs produced very good results on the set of training sentences for each of the sentence types. Thus, the SRNs were capable of recognizing the sentence structures *N V N*, *N V N who V N* and *N who N V V N* for each of the training sentences. Since the sentence structures in the test sentences were the same as in the training sentences, the SRNs would also recognize the sentence structures in the test sentences if the same information about sentence structure was presented to the networks in both cases. This will result when content words are first categorized as *Noun* or *Verb* before presentation to the networks, so that the networks would learn and recognize sentence structures on the level of *N V N*, *N V N who V N* and *N who N V V N* sequences.

When used in this way, however, SRNs can only be a part of a model of human language performance. Consider, for instance, the sentences *cat chases mouse* and *mouse chases cat*. Both sentences are *N V N* sentences and they are thus indistinguishable for the SRNs. Yet, these two sentences convey very different messages, and the purpose of language is to represent these differences. In other words, the ‘who does what to whom’ information in both sentences is very different, and humans can understand these differences. In particular, they can produce the correct answers to the ‘who does what to whom’ questions for each of these sentences (and the other sentences they understand). Such questions cannot be answered on the level of the *N V N* structure as represented by the SRNs.

This raises two important questions for the use of SRNs in models of this kind. First, how (in neural terms) is the difference between *cat chases mouse* and *mouse chases cat* represented in these models? This cannot be achieved with (another) SRN because of the combinatorial productivity that is needed for this form of representation. Second, how is the structural *N V N* information represented with the SRNs related with the ‘content’ representation for *cat chases mouse* and *mouse chases cat*? Basically, this is a ‘binding’ problem because it requires that, for instance, the first *Noun* in *N V N* is bound to *cat* in *cat chases mouse* and to *mouse* in *mouse chases cat*. But even if these problems can be solved, NLP on the basis of *N* and *V* sequences is faced with serious limitations. Consider the following sentences:

*The cat that the dog that the boy likes sees chases the mouse* (1)

*The fact that the mouse that the cat chases roars surprised the boy.* (2)

Both sentences have the same structure in terms of *Ns* and *Vs*: *N that N that N V V V N*. So, they would be indistinguishable for an SRN that processes sentences in terms of *N* and *V* sequences. Yet, there is a strong difference in complexity between both sentences (Gibson 1998). Sentence (1) is a sentence with a double centre-embedding, which is very hard to understand. Sentence (2) is a sentence with a complement clause, which can be understood reasonably well. The difference in complexity results from the difference in memory load during the processing of both sentences (Gibson 1998). In particular, in (1) the noun *cat* is both the object of the second verb *sees* and the subject of the third verb *chases*. In (2),

the noun *fact* is only the subject of the third verb *surprised*. These differences reflect the structural relations, or dependencies, that exist between nouns and verbs in a sentence, which cannot be represented in terms of  $N$  and  $V$  sequences.

In their review of the literature, Christiansen and Chater (2001) noted that (even after a decade of research) SRNs still use ‘toy’ fragments of grammar and small vocabularies. They concluded that more research is required to decide whether the models can scale up to the complexities of natural language. As we have argued here, a fundamental complexity of NLP is its combinatorial productivity, which results directly from its huge vocabulary, and which is not found in artificial languages. The results presented here illustrate, in our view, that the issue of combinatorial productivity should be at the forefront of NLP with connectionist systems.

### Acknowledgement

We wish to thank three anonymous reviewers for their valuable comments on this article.

### References

- Bloom, P., 2000, *How Children Learn the Meanings of Words* (Cambridge MA: MIT Press).
- Calvin, W. H., and Bickerton, D., 2000, *Lingua ex Machina: Reconciling Darwin and Chomsky with the Human Brain* (Cambridge MA: MIT Press).
- Chomsky, N., 1957, *Syntactic Structures* (The Hague: Mouton).
- Chomsky, N., 2000, *New Horizons in the Study of Language and Mind* (Cambridge: Cambridge University Press).
- Christiansen, M. H., and Chater, N., 1994, Generalization and connectionist language learning. *Mind & Language*, **9**: 273–287.
- Christiansen, M. H., and Chater, N., 1999a, Connectionist natural language processing: the state of the art. *Cognitive Science*, **23**: 417–437.
- Christiansen, M. H., and Chater, N., 1999b, Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, **23**: 157–205.
- Christiansen, M. H., and Chater, N., 2001, Connectionist psycholinguistics: capturing the empirical data. *Trends in Cognitive Sciences*, **5**: 82–88.
- Churchland, P. M., 1995, *The Engine of Reason, the Seat of the Soul* (Cambridge MA: MIT Press).
- Elman, J. L., 1991, Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, **7**: 195–225.
- Elman, J. L., 1995, Language as a dynamical system. In R. F. Port and T. van Gelder (eds) *Mind as Motion: Explorations in the Dynamics of Cognition* (Cambridge MA: MIT Press), pp. 195–223.
- Gers, F. A., and Schmidhuber, J., 2001, LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, **12**: 1333–1340.
- Gibson, E., 1998, Linguistic complexity: locality of syntactic dependencies. *Cognition*, **68**: 1–76.
- Grodzinsky, Y., 2000, The neurology of syntax: language use without Broca’s area. *Behavioral and Brain Sciences*, **23**: 1–71.
- Hadley, R. F., 1994, Systematicity in connectionist language learning. *Mind & Language*, **9**: 247–272.
- Ingram, D., 1989, *First Language Acquisition* (Cambridge: Cambridge University Press).
- Katz, N., Baker, E., and Macnamara, J., 1974, What’s in a name: a study of how children learn common and proper names. *Child Development*, **45**: 469–473.
- Marcus, G. F., 1998, Can connectionism save constructivism? *Cognition*, **66**: 153–182.
- Miikkulainen, R., 1996, Subsymbolic case-role analysis of sentences with embedded clauses. *Cognitive Science*, **20**: 47–73.
- Palmer-Brown, D., Tepper, J. A., and Powell, H. M., 2002, Connectionist natural language parsing. *Trends in Cognitive Sciences*, **6**: 437–442.
- Phillips, S., 1998, Are feedforward and recurrent networks systematic? Analysis and implications for a connectionist cognitive architecture. *Connection Science*, **10**: 137–160.
- Pinker, S., 1984, *Language Learnability and Language Development* (Cambridge MA: Harvard University Press).

Pinker, S., 1994, *The Language Instinct* (London: Penguin).

Pinker, S., 1998, *How the Mind Works* (London: Penguin).

Pulvermüller, F., 1999, Words in the brain's language. *Behavioral and Brain Sciences*, **22**: 253–336.

van der Velde, F., 1995, Symbol manipulation with neural networks: production of a context-free language using a modifiable working memory. *Connection Science*, **7**: 247–280.

## Appendix

### *Examples of sentences that were presented to a network during training*

Below are printed the first ten sentences of the training corpus that was fed to a network during the fourth phase of training. Note that sentences combine only nouns and verbs belonging to the same group of words, as distinguished in table 1.

1. *girl who boy sees hears girl.*
2. *Clarice follows Clarice.*
3. *Anthony loves Clarice.*
4. *boss dislikes clerk who obeys boss.*
5. *clerk who clerk obeys dislikes clerk.*
6. *girl who girl sees hears girl.*
7. *dog who cat chases chases cat.*
8. *Clarice follows Anthony who follows Anthony.*
9. *Clarice follows Anthony.*
10. *cat who cat kicks kicks cat.*

### *Examples of sentences that were presented to a network during testing*

Next to being tested on sentences combining only nouns and verbs belonging to the same group (like the sentences printed above), networks were also tested on sentences that combined, alternatively, nouns and verbs from more than one group. First, here are some examples of test sentences combining words from *two* different groups.

1. *clerk loves boss.*
2. *boy obeys girl.*
3. *clerk follows boss.*
4. *boy loves boy.*
5. *cat dislikes cat.*
6. *girl kicks boy.*
7. *dog hears dog who sees cat.*
8. *clerk hears boss who sees clerk.*
9. *girl loves girl who follows girl.*
10. *boy dislikes girl who obeys boy.*
11. *Anthony obeys Clarice who obeys Anthony.*
12. *clerk who dog dislikes chases boss.*
13. *Anthony who boss follows obeys Anthony.*
14. *cat who Anthony chases loves cat.*
15. *boy who Anthony sees loves girl.*

*Examples of some test sentences that combine words from three different groups follow.*

1. *girl kicks boss.*
2. *boy dislikes Anthony.*
3. *cat hears boss.*

4. *dog obeys boy.*
5. *boy obeys Clarice who hears boss.*
6. *cat dislikes Anthony who chases boss.*
7. *boss hears Anthony who dislikes girl.*
8. *Anthony sees clerk who loves girl.*
9. *boss who Clarice kicks dislikes Clarice.*
10. *dog who boy loves chases girl.*
11. *clerk who Anthony chases obeys Anthony.*
12. *girl who dog follows sees dog.*
13. *Clarice who clerk kicks follows clerk.*

Below are some examples of sentences that combine nouns and verbs from all *four* groups of words. Because simple sentences consist of only two nouns and one verb, these structures naturally cannot combine words from all four groups in one sentence. Therefore, only right-branching and centre-embedded sentences are found here.

1. *boss kicks Clarice who sees clerk.*
2. *Anthony dislikes girl who kicks Clarice.*
3. *girl follows dog who obeys girl.*
4. *boss follows dog who sees clerk.*
5. *boss who girl kicks loves clerk.*
6. *girl who Clarice chases dislikes boy.*
7. *clerk who cat follows sees boss.*
8. *cat who girl dislikes loves cat.*

*Examples of sentences that were presented to a network as a word-salad*

Finally, here are some examples of *seven-word-salads*, followed by a lexical item. These sequences were presented to the networks to obtain the (average) activations produced by each lexical item. These activations were used to determine the word–word associations produced by each lexical item.

1. *sees chases kicks dislikes follows boss Clarice **who***
2. *loves Clarice boss Clarice hears Anthony Clarice **who***
3. *dislikes boy dog clerk dog who dog **who***
4. *chases clerk Anthony obeys obeys . kicks **who***
5. ***Clarice** · dog girl dog clerk obeys **boy***
6. *dislikes clerk hears girl boy hears Anthony **boy***
7. *clerk girl dog boss obeys who dislikes **boy***
8. *dislikes loves . . Clarice obeys sees **boy***

*Examples of some sentences from the mixing group (figure 6)*

Test sentences:

1. *boy hears girl*
2. *dog sees cat*
3. *boy kicks girl who dislikes clerk*
4. *dog chases cat who obeys boss*

Training sentences:

1. *boy sees girl*
2. *dog hears cat*

3. *boy follows cat*
4. *dog loves girl*
5. *boy chases girl who dislikes clerk*
6. *dog kicks cat who loves Anthony*
7. *dog dislikes girl who follows Clarice*
8. *boy who cat hears obeys Clarice*
9. *girl who dog hears obeys clerk*
10. *boy who cat kicks loves Anthony*
11. *dog who girl chases follows Clarice*