Syntactic systematicity in sentence processing with a recurrent self-organizing network

Igor Farkaš *,1

Department of Applied Informatics, Comenius University
Mlynská dolina, 842 48 Bratislava, Slovak Republic

Matthew W. Crocker [2]

Department of Computational Linguistics, Saarland University
Saarbrücken, 66041, Germany

Abstract

As potential candidates for explaining human cognition, connectionist models of sentence processing must demonstrate their ability to behave systematically, generalizing from a small training set. It has recently been shown that simple recurrent networks and, to a greater extent, echo-state networks possess some ability to generalize in artificial language learning tasks. We investigate this capacity for a recently introduced model that consists of separately trained modules: a recursive self-organizing module for learning temporal context representations and a feed-forward two-layer perceptron module for next-word prediction. We show that the performance of this architecture is comparable with echo-state networks. Taken together, these results weaken the critisism of connectionist approaches, showing that various general recursive connectionist architectures share the potential of behaving systematically.

Key words: recurrent neural network, self-organization, next word prediction, systematicity

* *Email addresses:* farkas@fmph.uniba.sk (Igor Farkaš), crocker@coli.uni-sb.de (Matthew W. Crocker).

[1] Also part time with Institute of Measurement Science, Slovak Academy of Sciences, Bratislava. The work was supported in part by Slovak Grant Agency for Science and by the Humboldt Foundation.

[2] M. Crocker's research was supported by SFB 378 Project "Alpha", awarded by the German Research Foundation.

1 Introduction

The combinatorial systematicity of human language refers to the observation that a limited lexicon can be combined with a limited number of syntactic configurations to yield a very large, possibly infinite, number of possible sentences. As potential candidates for explaining human cognition, connectionist models must necessarily be able to account for the systematicity of human language. This potential capability was questioned by Fodor and Pylyshyn [10] and is still a matter of debate [4,1]. Hadley [13] first proposed that systematic behavior is a matter of learning and generalization: A neural network trained on a limited number of sentences should generalize to be able to process all possible sentences. Moreover, he claims, since people learn systematic language behavior from exposure to only a small fraction of possible sentences, a neural network should similarly be able to learn from a relatively small proportion of possible sentences, if it is to be considered cognitively plausible. Hadley further distinguishes betweeen weak and strong systematicity. A network is weakly systematic if it can process sentences with novel combinations of words, but these words are in the syntactic positions they also occurred in during training (e.g. the network trained on sentences *boy loves girl* and *dog chases cat* can also process *dog chases girl*). Strong systematicity, on the other hand, requires generalization to new syntactic positions (e.g. the ability to process the sentence *dog chases boy*, provided that noun *boy* never appeared as an object during training). [3]

According to Hadley, connectionist models were at best weakly systematic, whereas human language requires strong systematicity. Various connectionist attempts were restricted in various ways. Either they required specific representations [14] or network architectures [2], or they reported mixed results [5]. The most encouraging results with a general network architecture using a larger test set have been obtained in [12]. Nevertheless, what is still desired is the demonstration of robust, scalable, strong systematicity in various general connectionist models [11].

Van der Velde *etal*. [25] claimed that even weak systematicity lies beyond the capabilities of connectionist models. They evaluated a simple recurrent network (SRN, [6]) in an artificial language learning task (next-word prediction) and argued that their SRN failed to process novel sentences appropriately (e.g. by correctly distinguishing between nouns and verbs). However, Frank [11] extended these simulations and showed that even their SRN, whose limitations had arisen from overfitting in large networks [25], could display some gen-

[3] Bodén and van Gelder [3] proposed a more fine-grained taxonomy of the levels of systematicity, but since here we only focus on weak systematicity, there is no need to introduce this taxonomy here.

eralization performance if the lexicon size was increased. Furthermore, Frank demonstrated that generalization could be improved upon by employing an alternative architecture – the echo-state network (ESN, [16]) that requires less training (its input and recurrent weights are fixed) and is less prone to overfitting.

In our recent work, we investigated the potential benefit of an alternative approach based on self-organization, in learning temporal context representations. Specifically, these self-organizing modules based on Recursive SOM (RecSOM; [26]) learnt to topographically represent the most frequent subsequences (left contexts) from the input stream of symbols (English text). We experimented with various recursive self-organizing modules, coupled with two types of a single-layer prediction module [9]. Using a next-word prediction task we showed that the best performance was achieved by the so called RecSOM-sard module (to be explained in Sec. 3.1) coupled with a simple perceptron. This model also turned out to be more robust (with respect to node lesioning) and faster to train than SRNs. In this paper, we investigate the weak syntactic systematicity of the RecSOMsard-based model and compare its performance with ESN. [4]

2 Input data

The sentences were constructed using the grammar in Table 1, which subsumes the grammar used in [25]. The language consists of three sentence types: simple sentences with an N-V-N structure, and two types of complex sentences, namely, right-branching sentences with an N-V-N-w-V-N structure and centre-embedded sentences with an N-w-N-V-V-N structure ('w' stands for *who*). Complex sentence types represent commonly used English-like sentences such as *boy loves girl who walks dog* and *girl who boy loves walks dog*, respectively.

Table 1
Grammar used for generating training and test sentences.

$\mathsf{S} \rightarrow \mathsf{Simple}\ (.2) \mid \mathsf{Right}\ (.4) \mid \mathsf{Centre}\ (.4)$	$\mathsf{N} \rightarrow \mathsf{N}_1 \mid \mathsf{N}_2 \mid \mathsf{N}_3 \mid \mathsf{N}_4$
$\mathsf{Simple} \rightarrow \mathsf{N\ V\ N}\ .$	$\mathsf{V} \rightarrow \mathsf{V}_1 \mid \mathsf{V}_2 \mid \mathsf{V}_3 \mid \mathsf{V}_4$
$\mathsf{Right} \rightarrow \mathsf{N\ V\ N\ w\ V\ N}\ .$	$\mathsf{N}_x \rightarrow n_{x1} \mid n_{x2} \mid ...$
$\mathsf{Centre} \rightarrow \mathsf{N\ w\ N\ V\ V\ N}\ .$	$\mathsf{V}_x \rightarrow v_{x1} \mid v_{x2} \mid ...$

Content words (nouns and verbs) are divided into four groups $\mathrm{N}_1, ..., \mathrm{N}_4$ and $\mathrm{V}_1, ..., \mathrm{V}_4$. Let W denote the lexicon size (i.e. the total number of word types

[4] The shorter version of this work appeared in [8].

in the language), then each group has (W-2)/8 nouns and the same number of verbs. Hence, for W=18 we have four nouns and four verbs per group (*who* and '.' are also considered words). The training set consisted of all sentences in which all content words were taken from the same group. That is, simple sentences had the form "$n_{xi}\ v_{xj}\ n_{xk}$.", right branching sentences had the form "$n_{xi}\ v_{xj}\ n_{xk}$ w $v_{xl}\ n_{xm}$." and centre-embedded sentences had the form "n_{xi} w $n_{xj}\ v_{xk}\ v_{xl}\ n_{xm}$.". The range of indices depends on lexicon size W: $i, ..., m \in \{1, ...(W\text{-}2)/8\}$. The number of simple sentences used for training ranged from 32 ($W = 18$) to 500 ($W = 42$), and the number of complex sentences from 256 ($W = 18$) to 25000 ($W = 42$). Regarding the data set size, we followed the regime described in [11]. This controlled training setup ensured that the proportion of training sentences relative to all possible sentences remained very small ($\sim$0.4%) which is a linguistically motivated requirement.

In contrast to training sentences, each test sentence contained content words from as many different groups as possible, which corresponds to the most complicated case [25,11]. That is, each simple sentence had the form "$n_{xi}\ v_{yj}\ n_{zk}$.", where $x \neq y \neq z$. Analogically, the five content words in right branching and centre-embedded test sentences came from all four different groups. The number of simple test sentences ranged from 192 ($W = 18$) to 3000 ($W = 42$). To make testing more efficient, from the huge number of possible complex test sentences we randomly selected 100 right branching sentences and 100 centre-embedded sentences (similarly to [11]) for each lexicon size.

3 RecSOMsard-P2 model

Our architecture consists of two modules that can be trained separately: a context-learning RecSOMsard and a prediction module based on a two-layer perceptron (hence P2). Adding a hidden layer of units in the prediction module was shown to enhance prediction accuracy in ESN [11] and hence is also used here to facilitate comparison.

3.1 Model description

The architecture of the RecSOMsard module is shown in Figure 1a. It is based on RecSOM [26] that has an extra top layer appended to its output. Each RecSOM unit $i \in \{1, 2, ..., N\}$ has two weight vectors associated with it: $\mathbf{w}_i \in \mathcal{R}^W$ linked with an W-dimensional input $\mathbf{s}(t)$, and $\mathbf{c}_i \in \mathcal{R}^N$ linked with the context $\mathbf{y}(t-1) = (y_1(t-1), y_2(t-1), ..., y_N(t-1))$ containing map activations $y_i(t-1)$ from the previous time step.

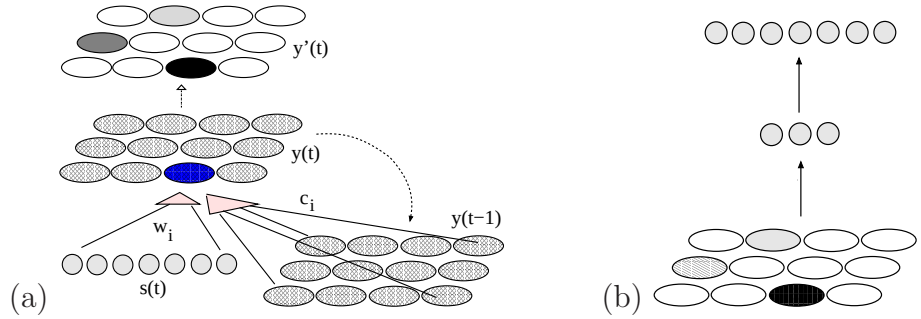

Fig. 1. (a) RecSOMsard architecture. The bottom part (without the top layer) represents RecSOM whose activity vector $\mathbf{y}$ is transformed to $\mathbf{y}$' by a mechanism described in the text. In RecSOM, solid lines represent trainable connections, and the dashed line represents a one-to-one copy of the activity vector $\mathbf{y}$. (b) A two-layer perceptron with inputs $\mathbf{y}$'.

The output of a unit i at time t is computed as $y_i(t) = \exp(-d_i(t))$, where

$$d_i(t) = \alpha \|\mathbf{s}(t) - \mathbf{w}_i\|^2 + \beta \|\mathbf{y}(t-1) - \mathbf{c}_i\|^2 \ . \tag{1}$$

In Eq. 1, $\|\cdot\|$ denotes the Euclidean norm, and parameters $\alpha > 0$ and $\beta > 0$ respectively influence the effect of the input and the context upon a unit's profile. Both weight vectors are updated using the same form of Hebbian learning rule [26]:

$$\Delta \mathbf{w}_i = \gamma . h_{ik}(t) . (\mathbf{s}(t) - \mathbf{w}_i) \tag{2}$$
$$\Delta \mathbf{c}_i = \gamma . h_{ik}(t) . (\mathbf{y}(t-1) - \mathbf{c}_i) \tag{3}$$

where k is an index of the winner, $k = \arg\min_{i \in \{1,2,\ldots,N\}} \{d_i(t)\}$ (which is equivalent to the unit with the highest activation $y_k(t)$), and $0 < \gamma < 1$ is the learning rate [26]. Neighborhood function h_{ik} is a gaussian (of width σ) on the distance $d(i,k)$ of units i and k in the map:

$$h_{ik}(t) = \exp\{-d(i,k)^2 / \sigma^2(t)\} \ . \tag{4}$$

The "neighborhood width", $\sigma(t)$, linearly decreases in time to allow for forming topographic representation of input sequences. RecSOM units self-organize their receptive fields to topographically represent temporal contexts (subsequences) in a Markovian manner [23]. However, unlike other recursive SOM-based models (overviewed in [15]), in case of a more complex symbolic sequence, RecSOM's topography of receptive fields can be broken which yields a non-Markovian fixed-input asymptotic dynamics [22,24].

The RecSOMsard module contains SardNet-like [17] output (untrained) post-processing whose output then feeds to a prediction module. In each iteration, the winner's activation y_k in RecSOM is transformed to a sharp Gaussian

profile $y'_i = \exp\{-d(i,k)^2/\sigma_y^2\}$ centered arround the winner k, and previous activations in the top layer are decayed via $\mathbf{y}' \leftarrow \lambda \mathbf{y}'$ as in SardNet. However, whereas SardNet assumes $\sigma_y^2 \approx 0$), for our prediction purposes, local spreading of the map activation (i.e. $\sigma_y^2 > 0$) turned out to be beneficial [9]. Once the winner is activated, it is removed from competition and cannot represent later input in the current sentence. It was observed in SardNet that forcing other (neighboring) units to participate in the representation allows each unit to represent different inputs depending on the context, which leads to an efficient representation of sentences, and which also helps to generalize well to new sentences. Hence, this feature is expected to transfer to RecSOMsard. At boundaries between sentences, all y'_i are reset to zero. Using the above procedure, the activations $\mathbf{y}(t)$ with mostly unimodal shape are transformed into a distributed activation vector $\mathbf{y}'(t)$ whose number of peaks grows with the position of a current word in a sentence. As a result, the context in RecSOMsard becomes represented both spatially (due to SardNet) and temporally (because RecSOM winner in the trained map best matches the current input in a particular temporal context). In [9] we concluded that this spatiotemporal representation of the context was the reason for the best performance of RecSOMsard.

3.2 Training the networks

Using localist encodings of words, networks were trained on the next word prediction task by being presented one word at a time. All training sentences were concatenated in random order. Following [11], the ratio of complex to simple sentences was 4:1 throughout the entire training phase, as it has been shown that "starting small" [7] is not necessary for successful SRN training [20]. For each model and each lexicon size, 10 networks were trained for ∼300,000 iterations and differed only in their initial weights, that were uniformly distributed between -0.1 and +0.1. First, the RecSOMsard module was trained, and then its outputs were used to train the perceptron. Values of some RecSOMsard parameters were found empirically and were then fixed: $\lambda = 0.9, \gamma = 0.1, \sigma_y = 1$. The effect of other parameters was systematically investigated (N, α, β). The perceptron, having 10 hidden units with logistic activation function (as in [11]), was trained by online back-propagation (without momentum), with the learning rate that was set to linearly decrease from 0.1 to 0.01. Cross-entropy was used as the error function, therefore the perceptron output units had softmax activation functions, i.e.

$$a_i = \frac{\mathrm{e}^{net_i}}{\sum_j \mathrm{e}^{net_j}}, \tag{5}$$

where net_j is the total input activation received by output unit j.

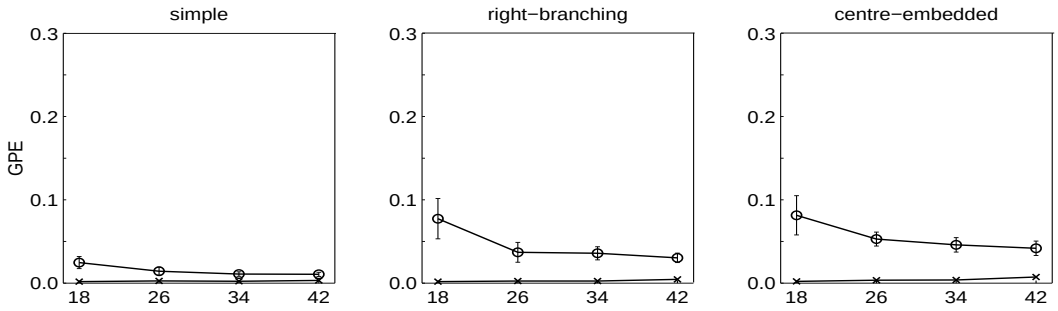

Fig. 2. Mean GPE measure for the three sentence types as a function of lexicon size W. Error bars were negligible for training data denoted by 'x' and hence are not shown. The lines marked with 'o' refer to the testing data.

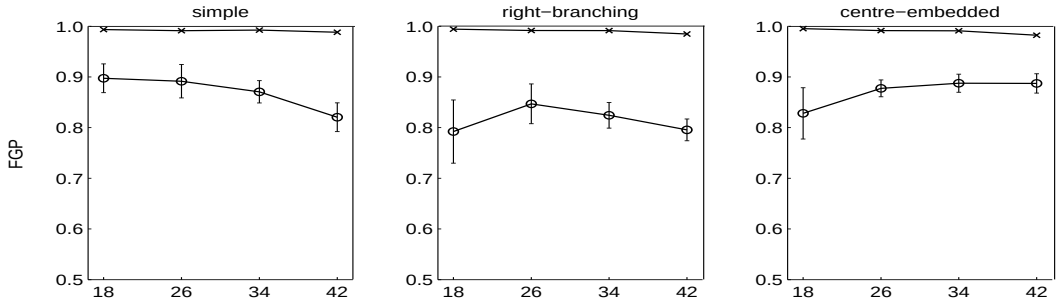

Fig. 3. Mean FGP measure for the three sentence types as a function of lexicon size W ('x' = training data, 'o' = testing data).

4 Experiments

4.1 Performance measures

We rated the network performance using two measures. Let us denote G the set of words in the lexicon that form grammatical continuations of the input sequence seen by the network so far. The first measure is the grammatical prediction error (GPE) defined in [25] as the ratio of the sum of non-grammatical output activations $a(\neg G) = \sum_{i \notin G} a_i$ and the sum of total output activation $a(\neg G) + a(G)$, i.e.

$$\text{GPE} = \frac{a(\neg G)}{a(\neg G) + a(G)} \tag{6}$$

Since we have softmax output units (i.e. the overall output activation is normalized to one [5]), GPE $= a(\neg G)$. Frank [11] argues that GPE lacks a baseline (that would correspond to the network with no generalization) and that this problem is overcome by an alternative measure he introduced to quantify the

[5] This does not hold, however, in case of using sigmoid output neurons, as in [25].

generalization (we call it Frank's generalization performance, FGP). FGP is based on comparing network's $a(G)$ with predictions of a bigram statistical model, whose grammatical activation $b(G) = \sum_{i \in G} b_i$, where b_i is the probability of the word i given the current word. FGP is formally defined as

$$\text{FGP} = \begin{cases} \frac{a(G)-b(G)}{b(G)} & \text{if } a(G) \leq b(G) \\ \frac{a(G)-b(G)}{1-b(G)} & \text{if } a(G) > b(G) \end{cases} \tag{7}$$

The marginal cases result as follows: If $a(G) = 1$, then FGP = 1 (perfect generalization); if $a(G) = 0$ (completely non-grammatical predictions), then FGP = -1; a non-generalizing network with $a(G) = b(G)$ (i.e. behaving as a bigram model) would yield FGP = 0. Hence, positive FGP score measures degree of generalization. As noted in [11], this scheme fails when $b(G) = 1$, 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 FGP = 0. With the given grammar, this only happens when predicting the beginning of the next sentence (which always starts with a noun). Therefore, network performance when processing '.' remains undefined.

4.2 Results

We ran extensive simulations that can be divided into three stages:
(1) *Effect of* N: First, we looked for a reasonable number of map units N (using map radii 9, 10, 12, 14, 16) and trying a few map paramater pairs (α, β) satisfying $1 < \alpha < 3$ and $0.4 < \beta < 1$. We found that beyond $N = 12 \times 12 = 144$ the test performance stopped to significantly improve. Hence, for all subsequent simulations we used this network size.
(2) *Effect of* α *and* β: We systematically investigated the influence of these map parameters and found that they did have an impact on the mean GPE. The following constraints were deduced to lead to the best performance: $0.8 \leq \alpha \leq 1.5$, $0.2 \leq \beta \leq 0.6$ and $0.4 \leq \alpha - \beta \leq 1.1$. Figure 2 shows the mean of mean GPEs (computed for 8 concrete α-β pairs from the above intervals) as a function of W. Training error was negligible, and test error that remains below 10% can be seen to reduce with larger lexicon. Similar dependence was observed in [11] in terms of increasing FGP, both in case of SRN and ESN models. In the case of our model (Figure 3), this trend in FGP is only visible for centre-embedded sentences. Also, our FGP values are slightly worse than those of ESN [11], but still clearly demonstrate the occurrence of generalization.

(3) *Mean word predictions:* Next, we took the model (given by N, α, β) with the lowest mean GPE and calculated its performance for individual inputs

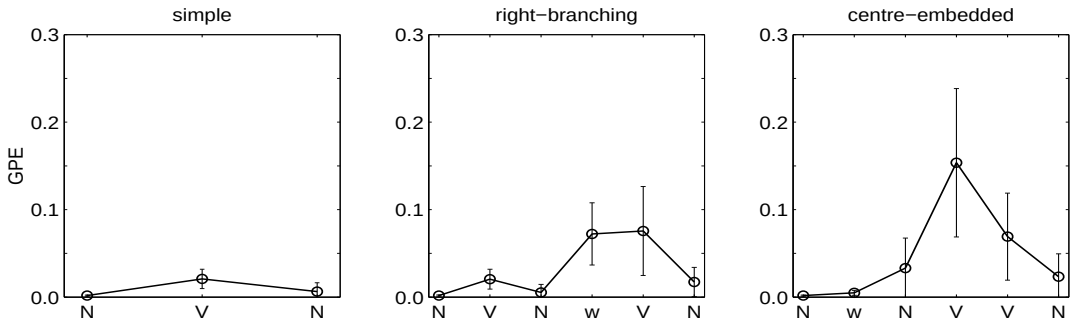

Fig. 4. Mean GPE for the three sentence types, averaged over test sentences and all lexicon sizes ($N = 144, \alpha = 0.8, \beta = 0.4$).

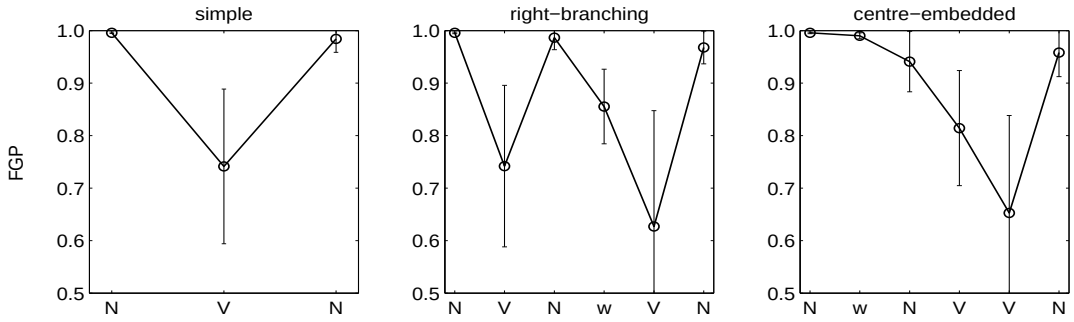

Fig. 5. Mean FGP for the three sentence types, averaged over test sentences and all lexicon sizes ($N = 144, \alpha = 0.8, \beta = 0.4$).

(words) in tested sentences, averaged over all lexicon sizes (W). Again, GPE on training data was close to 0. Results for test data are shown in Figure 4. The corresponding performance in terms of FGP in Figure 5 stays above 0.6. Compared to ESN ([11], Fig. 7), this performance falls between the best ESN model (for best W and N) whose FGP ≥ 0.8 and the mean FGP performance (averaged over W and N) which drops to 0.5 for the most difficult cases in complex sentences. In our model, the most difficult predictions in terms of both measures were similar for all word positions, and can be seen from Figures 4 and 5. Unlike ESN, our model predicted the end-of-sentence markers in case of complex sentences very well. To do so, the network has to have sufficient memory in order to learn the preceeding contexts (w-V-N and V-V-N, leading to '.'). In case of N-V-N context (that occurs both in simple and right branching sentences), the network correctly predicted both potential grammatical continutions '.' and 'w'). Both Figures 4 and 5 suggest that the RecSOMsard-P2 network is also capable of generalization. The two measures appear to be inversely related, but differ in the fact that only FGP depends on bigram performance (which could explain different peak positions in the two graphs for complex sentences with centre-embedding).

To illustrate the behaviour of our model, we chose one representative trained network ($W = 42, N = 144, \alpha = 0.8, \beta = 0.4$) and swept the testing set through it. In each iteration during the single epoch, we recorded RecSOMsard activations as well as P2 predictions of the trained model. Both data sets were

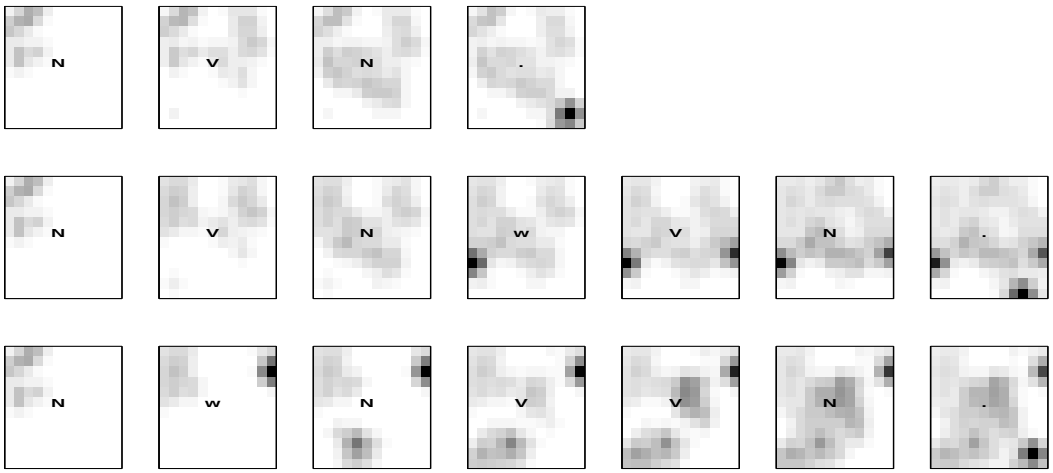

Fig. 6. Mean RecSOMsard activation (map 12×12) for three sentence types evaluated with a single representative model using the testing set. Symbols in the center show current input. For commentary see the text.

averaged with respect to corresponding word positions in the sentences (as also used in Fig. 4 and 5). Figure 6 shows the mean RecSOMsard activity (of 144 neurons) for all three sentence types, with the symbol in the centre showing the current input. These plots illustrate how RecSOMsard organizes its state space while reading the input sentences. Upon starting a new sentence, the map activations are zero, and with each incoming word, a new cluster of activations is added into the representation, while decaying the previous ones (SardNet-like mechanism). Due to frequent occurrence of symbols 'w' and '.', the activations associated with these inputs are most discernible in the map. Note that clusters of activation resulting from input 'w' spatially differ for right branching and centre-embedded sentences, hence allowing correct prediction of verbs and nouns, respectively. This is not the case of input '.', however, because all sentences start with a noun (as predicted by the model).

How these state-space representations lend themselves to generating next-word predictions is shown in the associated Figure 7. Each plot shows the mean prediction probabilities for four categories: '.', nouns (Ns), verbs (Vs) and 'w' at particular position within a sentence. The reason for grouping nouns and verbs is that since we focus on syntactic systematicity, there are no semantic constraints (allowed N-V combinations) and hence the network only needs to predict the correct syntactic category. As can be seen in Figure 7, in most cases the model correctly predicts the next four syntactic categories: In simple sentences, upon seeing the initial subject-noun, the network cannot know whether a simple or a centre-embedded sentence will follow, hence predicting both possibilities. Similar ambiguity in the grammar occurs at the end of the simple sentence with object-noun as the input. With right branching sentences, the most discernible inaccuracy in prediction is observed for

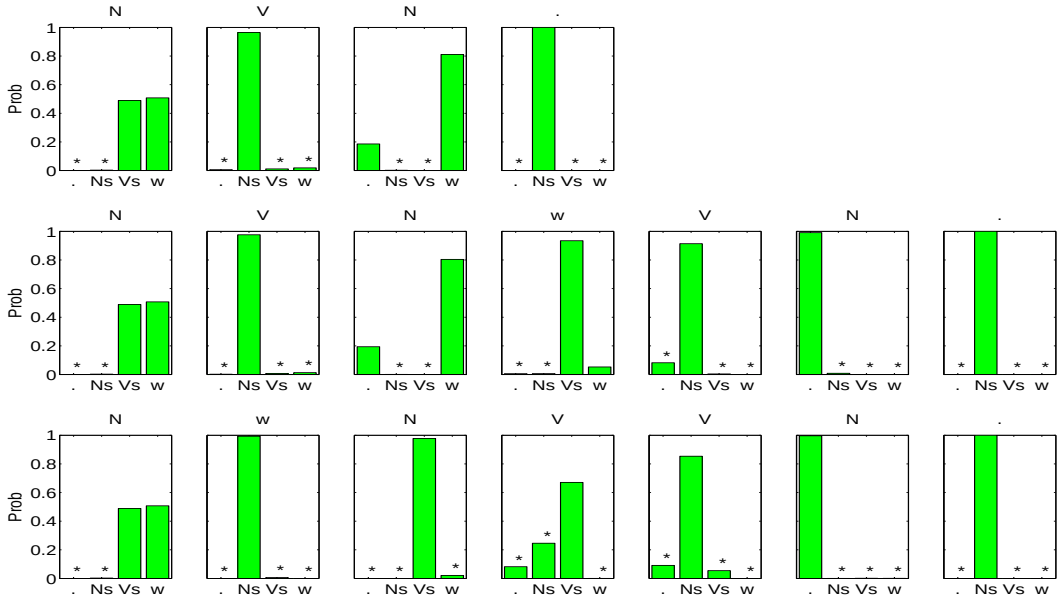

Fig. 7. Mean predictions for syntactic categories in case of three sentence types, evaluated with a single representative model on a testing set. Input symbols are shown above the figures. Non-grammatical predictions are labelled with $\star$.

input 'w' (when most ungrammatical prediction, 5.3% goes for '.') and for subsequent V (8.2% for '.'). This behaviour is consistent with right branching sentence plots in Fig. 4 and 5. Similarly in centre-embedded sentences, most inaccuracy can be seen for inputs V (8.2% for '.' and 24.6% for N) and for the next V (9.2% for '.' and 5.5% for V). Overall, the prediction accuracy can be considered very good, as the grammatical activation never drops below 67% for any word position.

5 Discussion

With regards to the criteria of weak systematicity, we have shown that RecSOMsard-P2, like ESN, largely avoids making non-grammatical predictions (quantified by GPE measure) which in turn indicates that the architecture displays some generalization (quantified by positive FGP). Since we achieved results comparable with ESN, it is a question whether in this task self-organization has its merits in learning context representations, as opposed to untrained weights used in ESN. On the other hand, although the performance of ESN comes at cheaper price, it is not clear whether using random (untrained) connections is biologically plausible, because the function of cortical circuits is typically

linked with self-organization [27]. [6]

The internal representations created by RecSOMsard output layer have the property of sparse codes, as a result of the SardNet property that distributes the representation of a sequence over the map [17]. This sparse code appears to be superior to the fully distributed codes formed in the hidden layer of SRN, as suggested by our node lesioning experiments: SRN exhibited a steeper performance degradation, compared to RecSOMsard, in the case of a similar next-word prediction task [9].

The next word prediction task is typically used in the context of connectionist language modeling [21]. It can be thought of as an inherent part of the language processor, although it does not (unlike some more complex sentence processing tasks, such as parsing) lead to formation of semantic representations of sentences that are assumed to be formed in human minds. However, it has been argued that language comprehension involves making simultaneous predictions at different linguistic levels and that these predictions are generated by the language production system [19]. This framework is in line with a general trend in cognitive sciences to incorporate action systems into perceptual systems and has broad implications for understanding the links between language production and comprehension. Hence, next word prediction appears to be an interesting approach since it permits a link between comprehension with production, albeit at higher level of abstraction. Comprehension in our model can be manifested by the model's current state space representation (RecSOMsard output) whose degree of accuracy predicts the accuracy of the next word token(s).

The presented architecture is not intended as a model of infant learning, but rather an investigation of how purely unbiased, distributional information can inform the learning the systematic syntactic knowledge in a variety of neural net architectures and training scenarios (SRN, ESN, RecSOMsard-P2). The use of symbolic (localist) rather than distributed word representations (that would contain syntactic and/or semantic features) is justified by the claim [18] that connectionist models, as a qualitatively different cognitive architecture, want to avoid the distinction between word tokens (lexicon) and syntactic word categories (expressed in terms of abstract rules of grammar). Therefore, connectionist models should operate directly on word tokens and try to learn grammar from these. Learning the grammar purely from co-occurrences between arbitrarily coded words (such as localist) is a more difficult task than using additional syntactic and/or semantic features in word representations, which would lead to the simplification of learning, because the network could

[6] We admit that this argument is weakened in our model because it uses backpropagation learning. Even in the case of a single-layer prediction (P) module without backpropagation (as in [9]), however, we obtained some degree of generalization.

take advantage of this information.

In conclusion, our results indicate that systematic behavior can be observed in a variety of connectionist architectures, including that presented here. Our findings thus further weaken the claim made by Fodor and Pylyshyn (or some of their supporters) that even if you find one example of connectionist systematicity, it does not really count because connectionism should be systematic "in general" to be taken seriously as a cognitive model. Investigating the learning ability from distributional information is a prerequisite to developing more cognitively faithful connectionist models, such as of child language acquisition.

Acknowledgment

We are thankful to three anonymous reviewers for their useful comments.

References

[1] K. Aizawa, The Systematicity Arguments, Kluwer Academic, Dordrecht, 2003.

[2] M. Bodén, Generalization by symbolic abstraction in cascaded recurrent networks, Neurocomputing 57 (2004) 87–104.

[3] M. Bodén, T. van Gelder, On being systematically connectionist, Mind and Language 9 (3) (1994) 288–302.

[4] D. Chalmers, Connectionism and compositionality: Why Fodor and Pylyshyn were wrong, Philosophical Psychology 6 (1993) 305–319.

[5] M. Christiansen, N. Chater, Generalization and connectionist language learning, Mind and Language 9 (1994) 273–287.

[6] J. Elman, Finding structure in time, Cognitive Science 14 (1990) 179–211.

[7] J. Elman, Learning and development in neural networks: The importance of starting small, Cognition 48 (1) (1993) 71–79.

[8] I. Farkaš, M. Crocker, Systematicity in sentence processing with a recursive self-organizing neural network, in: Proceedings of the 15th European Symposium on Artificial Neural Networks, 2007.

[9] I. Farkaš, M. Crocker, Recurrent networks and natural language: exploiting self-organization, in: Proceedings of the 28th Annual Conference of the Cognitive Science Society, Lawrence Erlbaum, Hillsdale, NJ, 2006.

[10] J. Fodor, Z. Pylyshyn, Connectionism and cognitive architecture: A critical analysis, Cognition 28 (1988) 3–71.

[11] S. Frank, Learn more by training less: systematicity in sentence processing by recurrent networks, Connection science 18 (3) (2006) 287–302.

[12] S. Frank, Strong systematicity in sentence processing by an echo-state network, in: Proceedings of ICANN, Part I, Lecture Notes in Computer Science, vol. 4131, Springer, 2006.

[13] R. Hadley, Systematicity in connectionist language learning, Mind and Language 9 (3) (1994) 247–272.

[14] R. Hadley, A. Rotaru-Varga, D. Arnold, V. Cardei, Syntactic systematicity arising from semantic predictions in a hebbian-competitive network, Connection Science 13 (2001) 73–94.

[15] B. Hammer, A. Micheli, A. Sperduti, M. Strickert, Recursive self-organizing network models, Neural Networks 17 (8-9) (2004) 1061–1085.

[16] H. Jaeger, Adaptive nonlinear system identification with echo state networks, in: Advances in Neural Information Processing Systems 15, MIT Press, Cambridge, MA, 2003.

[17] D. James, R. Miikkulainen, Sardnet: a self-organizing feature map for sequences, in: Advances in Neural Information Processing Systems 7, MIT Press, 1995.

[18] R. Miikkulainen, Subsymbolic case-role analysis of sentences with embedded clauses, Cognitive Science 20 (1996) 47–73.

[19] M. Pickering, S. Garrod, Do people use language production to make predictions during comprehension?, Trends in Cognitive Sciences 11 (2007) 105–110.

[20] D. Rohde, D. Plaut, Language acquisition in the absence of explicit negative evidence: How important is starting small?, Cognition 72 (1999) 67–109.

[21] D. Rohde, D. Plaut, Connectionist models of language processing, Cognitive Studies 10 (1) (2003) 10–28.

[22] P. Tiňo, I. Farkaš, On non-markovian topographic organization of receptive fields in recursive self-organizing map, in: L. Wang, K. Chen, Y. Ong. (eds.), Advances in Natural Computation – ICNC 2005, Lecture Notes in Computer Science, Springer, 2005.

[23] P. Tiňo, I. Farkaš, J. van Mourik, Recursive self-organizing map as a contractive iterative function system, in: M. Gallagher, J. Hogan, F. Maire (eds.), Intelligent Data Engineering and Automated Learning – IDEAL 2005, Lecture Notes in Computer Science, Springer, 2005.

[24] P. Tiňo, I. Farkaš, J. van Mourik, Dynamics and topographic organization in recursive self-organizing map, Neural Computation 18 (2006) 2529–2567.

[25] F. van der Velde, G. van der Voort van der Kleij, M. de Kamps, Lack of combinatorial productivity in language processing with simple recurrent networks, Connection Science 16 (1) (2004) 21–46.

[26] T. Voegtlin, Recursive self-organizing maps, Neural Networks 15 (8-9) (2002) 979–992.

[27] C. von der Malsburg, Self-organization and the brain, in: M. Arbib (ed.), The Handbook of Brain Theory and Neural Networks, MIT Press, 2003, pp. 1002–1005.