INVESTIGATING SYSTEMATICITY IN THE LINEAR RAAM NEURAL NETWORK

I. FARKAŠ* and M. POKORNÝ

Comenius University
Mlynská dolina, 84248 Bratislava, Slovak Republic
**E-mail: farkas@ii.fmph.uniba.sk*

Processing structured data is a continuing challenge for connectionist models that aim at becoming a plausible explanation of human cognition. The recently proposed linear Recursive Auto-Associative Memory (RAAM) model was shown to have a much higher encoding capacity and not to be subject to over-training compared to classical RAAM. We assess the effect of terminal encoding on the performance of linear RAAM in case of encoding trees of ternary semantic propositions and we show that the highest representation capacity is achieved with (sparse) binary WordNet-based codes, compared to (symbolic) neutral and to (distributed) word co-occurrence based codes. Only with WordNet codes the model could generalize to processing structures that contain known words at new syntactic positions or contain novel words, as long as these shared semantic features with the words from the training set.

Keywords: Syntactic systematicity; Linear recursive autoassociative memory; Neural network; Word features

1. Introduction

In their fundamental criticism in 1988, Fodor and Pylyshyn[1] expressed their serious doubts regarding connectionist models of that time; namely, whether they could account for generativity and systematicity observed in mental representations without merely implementing symbolic systems. Generativity expresses the idea that mental representations can be generated in an unlimited way, by combinatorial manipulations of atoms. Systematicity refers to the mental property that understanding certain sentences (e.g. `John loves Mary`) inherently implies understanding of related sentences (such as `Mary loves John`). This concept was more clearly defined by Hadley[2] who proposed that systematic behaviour in connectionist network was a matter of learning and generalization. Hadley distinguished three levels of systematicity: weak, quasi-, and strong. Niklasson & van

Gelder[3] subsequently proposed a more comprehensive and more detailed taxonomy (levels 0 through 5, with increasing "degree" of novelty in testing sentences), having loose correspondence to Hadley's three levels. Connectionist models, as a qualitatively different cognitive architecture, were challenged in showing they could represent structured data without pointers or logical addresses (natural part of symbolic systems), using vectors of fixed dimension. Since 1990 we have witnessed a number of connectionist attempts to handle systematicity,[22] out of which Recursive Auto-Associative Memory[5] appearantly attracted most attention, reflected in a variety of applications and modifications of the original RAAM.[3,6–8,10,11,13] The RAAM, as a recursive auto-encoder trained by error backprogation, learns the compressed (reduced) representations at its hidden layer. Despite its widespread use among connectionists, RAAM is known to have a number of drawbacks: the difficulty to train, sensitivity to noise and a rather poor generalization, to name a few.[9] The most recent alternative to the original model – linear RAAM by Voegtlin and Dominey (henceforth, V&D),[14] was reported to achieve a much better generalization performance and to avoid the problem of overtraining. In this paper, we examine the linear RAAM and illustrate its properties in the context of testing its systematicity in processing linguistic data using various encoding schemes of terminals (words).

2. Linear RAAM

The recently proposed linear RAAM[14] differs from original RAAM[5] in three points: (1) it uses neurons with linear activation function (identity mapping), as opposed to sigmoidal neurons, (2) it uses unsupervised Oja's rule for updating weights, unlike supervised error backpropagation, and (3) it uses the same weight matrix for both encoding and decoding structures (due to linearity).

Like RAAM, linear RAAM is a three-layer neural network that learns to encode n-ary trees (using nk-k-nk units). The tree encoding proceeds recursively bottom-up and left-right which is illustrated foor a binary tree in Figure 1a. First, we obtain reduced representations of subtrees `(A B)` and `(C D)` by presenting these pairs as inputs one at a time. The obtained representation of `(C D)` is then copied via recurrent links from the middle layer (MID) to the left group of inputs (LI). The right group of inputs (RI) reads the symbol `E`, and the mapping yields the representation of the subtree `((C D) E)`. The last step consists in copying this representation to RI and reading the previously obtained subtree representation of `(A B)` to LI. The activity of MID will then correspond to the reduced representation

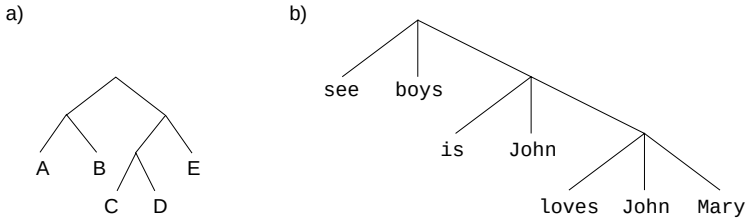

Fig. 1. Example of (a) binary and (b) ternary tree structure. The binary tree has leaves without any meaning. Terminals on the right are words that bear linguistic meaning, which can encoded in the terminal representations.

of the whole tree.

If we denote $z_j^{(a)}$ the activity of neuron j from the group a (out of n) in the input layer, c_i the activity of the neuron i of the compressing layer, and $w_{ij}^{(a)}$ the synaptic weight between these two neurons, then we can express encoded activations as

$$c_i = \sum_{a=1}^{n} \sum_{j=1}^{k} w_{ij}^{(a)} z_j^{(a)} \tag{1}$$

or, in the vector form as $\mathbf{c} = \mathbf{W}^{(1)}\mathbf{z}^{(1)} + \mathbf{W}^{(2)}\mathbf{z}^{(2)} + \ldots + \mathbf{W}^{(n)}\mathbf{z}^{(n)} = \mathbf{W}\mathbf{z}$, where $\mathbf{z}^{(a)}$ is a k-dimensional column vector representing neuron activations in input group a, $\mathbf{c}$ is the activation vector of MID and $\mathbf{W}^{(a)} = \|w_{ij}^{(a)}\|_{k\times k}$ is the weight matrix between a-th group of inputs and MID. Then, the whole weight matrix $\mathbf{W} = [\mathbf{W}^{(1)}; \mathbf{W}^{(2)}; \ldots \mathbf{W}^{(n)}] \in \mathbb{R}^{k\times nk}$, and $\mathbf{z} = [{\mathbf{z}^{(1)}}^T; {\mathbf{z}^{(2)}}^T; \ldots; {\mathbf{z}^{(n)}}^T]^T \in \mathbb{R}^{nk\times 1}$ is the vector of input activations.

2.1. *Decoding process*

Reconstruction of the original tree also proceeds recursively, but in top-down fashion which is illustrated using the ternary structure in Figure 1b. First, we copy the representation of the whole tree (corresponding to the sentence `boys see John who loves Mary`) to MID. As a result of its decomposition we obtain the representation of the terminal `see` in LI and the representation of the terminal `boys` in the middle group of inputs (MI). The activation pattern in RI does not correspond to any of the terminals, so its contents is copied to MID and decoded. As a result, now we get in LI and MI the representations of terminals `is` and `John`, respectively. Since the contents of RI does not correspond to any of the terminals, it is copied to MID and decoded, which results in decoded representations of all three terminals at the correponding groups of inputs, namely, `loves`, `John` and

`Mary`. In general, trees can have at certain depth more vertices that are not leaves. For example, both children of the root in Figure 1a are not leaves. In this case, during encoding we must store the obtained representation of a subtree in a stack, and later retrieve it and use it as input, after we have obtained the representations of the remaining subtrees. Analogical use of the stack is available during decoding the structure. For decoding of the group activity we use the same weight matrix, i.e. $\bar{\mathbf{z}}^{(a)} = \mathbf{W}^{(a)^T}\mathbf{c}$ and the overall mapping from MID to the output layer is $\bar{\mathbf{z}} = \mathbf{W}^T\mathbf{c}$. However, the decoding process is rarely ideal in a sense that the reconstructed images $\bar{\mathbf{z}}$ would exactly match the originals. [a] Therefore, we need a terminal test that could help us decide whether the obtained representation is to be considered a terminal or it should be further decoded.

2.2. *Terminal test*

In terminal test used by Pollack[5] the reconstructed vector was considered a terminal, if all its elements differed by less than τ from the required values, where he used $\tau = 0.2$. V&D[14] used the Euclidean distance instead, and a decoded vector was considered to encode a terminal if and only if the Euclidean distance to the vector encoding this terminal was below a *reconstruction threshold* θ. We need to specify what we mean by successful decoding. Even though we can allow certain inaccuracies during decoding while applying the terminal test, we may not succeed in reconstructing the original structure. The following cases can result: (i) ambiguous case — if for chosen θ, the reconstructed vector could represent more than one terminal; (ii) unrecognized terminal — if the reconstructed structure should yield a terminal but it does not (which could lead to infinite loops in the decoding process); (iii) terminating non-terminal — if at certain position we decode a terminal but the original structure contains a subtree at that position; (iv) wrong terminal decoded — if that differs from the required terminal. We will consider *decoding the structure* successful, if none of the above cases occurs. Such a structure will be considered encodable (representable) by the network.

[a] Formation of reduced representations involves dimensionality reduction. The rank of the matrix $\mathbf{P} = \mathbf{W}^T\mathbf{W}$ is $rank(\mathbf{P}) \leq k$, whereas the dimension of input vectors is nk. Therefore, if the input contains more than k linearly independent vectors, their reconstructions (using mapping defined by $\mathbf{P}$) will be linearly dependent vectors and will hence differ from the originals.

2.3. *Training the network*

The network si trained as an autoassociator. Minimization of the quadratic error between input (target) $\mathbf{z}$ and its reconstruction $\bar{\mathbf{z}}$ yields the stochastic rule for weight modification

$$\begin{array}{l}\forall 1 \leq i,j \leq k \\ \forall 1 \leq a \leq n\end{array} \qquad \Delta w_{ij}^{(a)} = \eta\, c_i \left(z_j^{(a)} - \sum_{r=1}^{k} w_{rj}^{(a)} c_r \right) \tag{2}$$

where η is the learning rate. This corresponds to Oja's constrained Hebbian learning rule,[15] which is known to find the linear subspace spanned by k principal components of the distribution of an input vector. Principal Components Analysis allows to linearly transform data from high-dimensional input space to the feature space of lower dimension. It consists in finding orthogonal vectors corresponding to the directions with the highest variance. The linear RAAM performs a more complex operation than does PCA, because the input vector $\mathbf{z}(t)$ depends on reduced representation $\mathbf{c}(t-1)$, which in turn depends on previous input $\mathbf{z}(t-1)$. Hence, the distribution of a vector $\mathbf{z}$ is not defined a priori, but it results from the internal representation devised by the network. This type of learning is called a *moving target problem*.

The adaptation runs recursively bottom up. First, the network is presented only parts of the structure that contain leaves. We always remember the representation of the presented subtree and modify weights according to Eq. 2. In the next steps, we process those parts of the structure that contain a subtree as well. This process is illustrated in Table 1 referring to the tree in Figure 1a.

Table 1. The order of training inputs for the structure in Figure 1a.

Input $\mathbf{z}$		Reduced repr.		Output $\bar{\mathbf{z}}$
(A B)	$\longrightarrow$	$R_{AB}(t_1)$	$\longrightarrow$	($\bar{\text{A}}$ $\bar{\text{B}}$)
(C D)	$\longrightarrow$	$R_{CD}(t_2)$	$\longrightarrow$	($\bar{\text{C}}$ $\bar{\text{D}}$)
($R_{CD}(t_2)$ E)	$\longrightarrow$	$R_{CDE}(t_3)$	$\longrightarrow$	($\bar{R}_{CD}(t_2)$ $\bar{\text{E}}$)
($R_{AB}(t_1)$ $R_{CDE}(t_3)$)	$\longrightarrow$	$R_{ABCDE}(t_4)$	$\longrightarrow$	($\bar{R}_{AB}(t_1)$ $\bar{R}_{CDE}(t_3)$)

3. Simulation experiment

3.1. *Encoding schemes for terminals*

To assess the effect of terminal encoding on network performance, we compared three types of features for the terminals – words in our linguistic

task: (a) the (symbolic) neutral code, (b) word encoding derived from word co-occurrences, and (c) word encoding containing WordNet-based features. For input data we generated English sentences based on specified probabilistic context-free grammar, using semantic constraints.[17] 300 sentences, based on the lexicon of 50 words, were transformed into ternary trees of propositions. Table 2 shows a few examples of simpler generated sentences and their translations. [b] In sentence translation, we tried to preserve, following Pollack, the recursive order (ACTION AGENT OBJECT), where these categories correspond to verb, subject, and object, respectively. In sentences with the missing object, we used a new terminal `NULL`. Note that ternary structures do not include words `who` and `who_pl`.

Table 2. Examples of simpler generated sentences and their translations.

Steve walks	(walks Steve NULL)
women see boys	(see women boys)
dogs who_pl see girl bark	(bark (are dogs (see dogs girl)) NULL)
boy feeds cat who John sees	(feeds boy (is cat (sees John cat)))

Note: For a sentence with embedding of depth n, the tree depth is $2n + 1$.

3.2. *Generation of word features*

The neutral codes implied 50-dimensional localist (one-hot) word representations, and the special symbol `NULL` was representated by the zero vector. Hence, the distance between any two meaningful terminals was $\sqrt{2}$, but the distance between `NULL` and any of these terminals was 1. Therefore, the optimal $\theta_{opt} > 0.5$. Word co-occurrence-based word encoding were created using the a special recurrent neural network – word co-occurrence detector (WCD) that learns the lexical co-occurrence constraints of words.[18,19] WCD reads through a stream of input sentences (one word at a time) and learns the transitional probabilities between words (the window size can be modulated, in this work we considered two nearest neighbors on either side) which it represents as a matrix of weights. Given a total lexicon of size N, all word co-occurrences can be represented by an $N \times N$ contingency table, where the representation for the ith word is formed by concatenation of ith column and ith row vectors from the table. For symbols `is` and

[b] Symbol who_pl was introduced for translation purposes, to differentiate between the singular and the plural.

`are` we used WCD codes of `who` and `who_pl`, respectively. Investigation of the WCD data revealed, that the minimal distance betwen two words was $d_{min} < 0.05$, so to enhance word discrimination, we rescaled the vectors such that its maximum component reached activation one. Since the largest vector component for some words was almost 0.5, the rescaling of the vector components was done using factor of two. In this case we estimated the optimal $\theta_{opt} \in (0.05, 0.1)$.

WordNet-based word features were derived by a feature generation system[20] that can produce a (smaller) set of binary features for each word. Harm's software incorporates semantic features mainly from WordNet,[21] but it did not provide features for all words so we generated those manually. Altogether, we had 41 features for all 50 words, so the word representations could be were 41-dimensional. Each word (except `NULL`) had 1 to 6 features. For this encoding, the minimum Euclidean distance of two words was $d_{min} = 1$, so the optimal threshold $\theta_{opt} > 0.5$.

The minimum network size that would accommodate all three types of word representation would be $k = 100$ (for neutral and WordNet codes the remaining bits would be padded with zeros). Preliminary simulations showed, however, that the network capacity could be increased if we provided extra room for encoding structures, so we used $k = 150$. For training, we generated 100 sentences, resulting in the overall amount of 325 ternary structures. In order to achieve successful training, we had to use a rather small learning η between 10^{-6} and 10^{-7} (for larger η, the weights diverged). As an undesirable consequence, the training time became rather extensive, especially for WCD codes, because even after 200000 iterations, the weights were still not converged.

3.3. *Levels of systematicity*

In the original paper of Fodor and Pylyshyn, the use of concept of systematicity was rather vague, so it was difficult to test it in connectionist models. In our experiments, we followed the taxonomy proposed by Niklasson & van Gelder,[3] shown in Table 3. For training we used in all experiments 150 structures from the base, but the actual selection of training sentences depended on the level of systematicity being tested. In the subsequent plot, we show results averaged over 7 runs. For each run, the weights were initialized to small values randomly chosen $[-0.1; 0.1]$. For neutral and WordNet codes we set $\eta = 0.001$ and for WCD codes $\eta = 005$. In search for optimal reconstruction, we set $\theta_{\mathrm{neutral}} = 0.55$, $\theta_{\mathrm{WCD}} = 0.1$, and $\theta_{\mathrm{WordNet}} = 0.7$. In all cases, the training was stopped after 5000 epochs.

Table 3. Levels of systematicity tested with linear RAAM.

Level	Description of the test set
0	No novelty (training data used)
1	Novel sentences (novel word combinations)
2	Novel positions (of at least one atom)
3	Novel atoms (at least one atom never appeared in training)
4	Novel complexity (of test sentences compared to training)
5	Novel atoms and novel complexity (combination of 3 and 4)

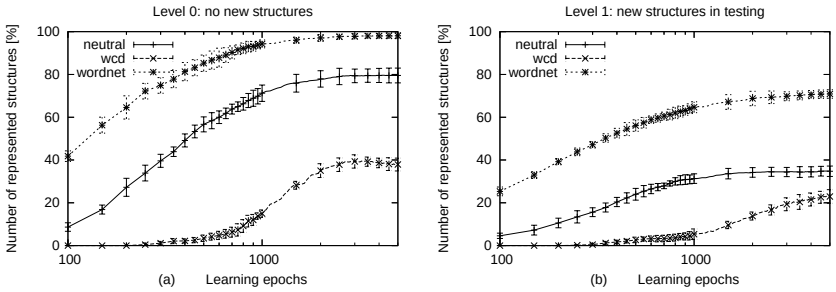

Fig. 2. Systematicity performance for (a) the training set (b) the testing set (level 1).

Figure 2a shows the results regarding systematicity level 0, i.e. using training data. In case of WordNet code, the network could represent almost all 150 structures from the training set, in case of neutral code it was approximately 80% and in case of WCD code somewhat below 40%.

As a next step, the networks were presented novel structures from the test set (level 1). Results are in Figure 2b. In case of neutral code, the performance of the network dropped to roughly 40% of novel structures in the test set (from 80% in the training set). If we assume that the network capacity is sufficient for representing 120 structures (which is 80% out of 150), then the network learnt only a half of novel structures. Analogical argument in case of WCD and WordNet codes leads to the conclusion that the network only learnt to exploit roughly 75% of its representational capacity.

For testing level 2 we correspondingly selected propositions for the test set: Since only nouns can occur at more than one syntactic position in our propositions, we excluded a few of these from the object position: `John` and `Steve`, singular `girl`, and plural `dogs`. The results are shown in Figure 3a. With the neutral code (not shown), the network failed to represent a single novel structure. For WCD and WordNet codes, the network achieved a certain degree of systematicity, albeit not impressive and highly

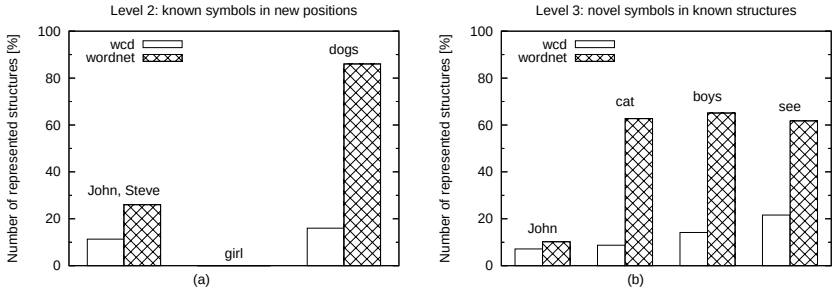

Fig. 3. Test results for higher levels of systematicity.

atom-dependent. The best result was observed for structures containing the atom `dogs` in the object position. In case of `John` and `Steve` we had to use a different thresholds ($\theta_{\mathrm{WCD}} = 0.07$, $\theta_{\mathrm{WordNet}} = 0.9$) to achieve the accuracy shown in Figure 3a. Finally, in case of `girl` the network failed to represent a single novel structure, irrespective of the word codes. The reasons for differences in behavior can be found in the test set itself. The test set for case `girl` only contained this atom at the lowest levels (at least 3) of the trees, so one could expect that the error during reconstructing the novel word will be higher compared to other words that appeared at the same positions in both training and test sets. If the word appeared at larger depth, the error is accumulated in reduced representations of corresponding superstructures. At the same time, each additional level probably increases the inaccuracy in decoding a superstructure. Therefore, we got zero performance in case of `girl`. This hypothesis is supported by the existence of represented structures in case of `John` and `Steve`: these atoms always occurred at depth 1.

For testing level 3, we prepared four data sets for training (and testing), each having one word (noun or verb) excluded from all training structures, but contained in all test structures. These words were: `John`, `cat`, `boys` and `see`. The average results are displayed in Figure 3b. For `John` we used $\theta_{\mathrm{WCD}} = 0.07$ and $\theta_{\mathrm{WordNet}} = 0.9$. To make the network able to represent structures containing a novel word there must exist a "very similar" word to it (in terms of Euclidean distance) in the training set. Hence, in case of neutral code we got zero performance for this level, whereas in case of WordNet code there exist pairs of words differing only in one feature, such as `cat` and `cats`. Note that in case of WordNet code, the performance for `John` is much worse compared to other words. The reason is that its closest neighbor, `Steve` (distance $\sqrt{2}$), only occurred in 10 out of 150 structures

from the training set, so `John`, unlike other three cases (`cat`, `boys` and `see`) did not have "sufficient training" to become represented during testing. Finally, regarding the level 4 we found that the network was unable to represent deeper structures, irrespective of the three word encodings used. We assumed that level 5 would also be beyond the representatinal capacity of the linear RAAM.

To summarize, the linear RAAM using neutral code can only satisfy level 1 systematicity, which is similar to what was concluded by V&D [14, sec. 6.5]. In case of WCD and WordNet codes we could observe a certain degree (higher for WordNet) of systematic behavior at levels 2 and 3, which depends on two factors: First, due to expected inaccuracy in reconstructing the word, i.e. during testing novel word [c], the reconstruction error is more problematic if the word lies at larger depth within a structure. Second, if the training set contains a word with a code similar (in terms of Euclidean distance) to the novel word, then the trained network is capable of representing also structures containing that novel word, thank to exploiting shared word features.

4. Conclusion

Our experiments with the recently proposed linear RAAM[14] shed some light on the network learning and representation properties. We investigated the effect of terminal encoding by comparing neutral code with two types of semantic features expecting that these would boost the representation capacity of the network. One type of features involved word encoding extracted from word co-occurrences within the English-like text corpus (context-free grammar with superimposed semantic constraints), the other was based on WordNet database. In the task of encoding trees of ternary semantic propositions, we observed that clearly the highest representation capacity was achieved with (sparse) binary WordNet codes. On the other hand, despite the appeal of word co-occurence models in linguistic modeling,[23] the word co-occurrence features did not work well in our task, because although this approach displayed the lowest reconstruction error, the real-valued features led to a high number of confusions in decoding, given the considered Euclidean metrics. In the context of testing levels of systematicity in the linear RAAM, according to taxonomy proposed by Niklasson & van Gelder,[3] we showed that the network with WordNet-based terminal en-

[c] such as being a known atom at novel position (level 2), or as a completely novel atom (level 3)

coding accomplished to the certain degree the third level of systematicity, i.e. it could generalize to processing structures that contain known words at new syntactic positions or novel words, as long as these shared semantic features with the words from the training set. In summary, these results suggest that the linear RAAM has remarkable generalization properties, and that connectionist representing symbolic structures (i.e. using neutral code) has its soft limits. In language domain, the linear RAAM was shown to benefit from using appropriate binary semantic features of terminals in the process of generalization.

5. Acknowledgments

This work was suported by Slovak Grant Agency for Science (VEGA, no. 1/0361/08).

References

1. J. Fodor and Z. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71, 1988.
2. R. Hadley. Systematicity in connectionist language learning. *Mind and Language*, 9(3):247–272, 1994.
3. L. Niklasson and T. van Gelder. On being systematically connectionist. *Mind and Language*, 9(3):288–302, 1994.
4. G. Hinton. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2):47–75, 1990.
5. J. Pollack. Recursive distributed representations. *Artificial Intelligence*, 46(1-2):77–105, 1990.
6. D. Chalmers. Syntactic transformations on distributed representations. *Connection Science*, pages 46–55, 1990.
7. L. Chrisman. Learning recursive distributed representations for holistic computation. *Connection Science*, 3:345–366, 1991.
8. R. Miikkulainen. Subsymbolic case-role analysis of sentences with embedded clauses. *Cognitive Science*, 20:47–73, 1996.
9. D. Blank, L. Meeden, and J. Marshall. Exploring the symbolic/subsymbolic continuum: A case study of RAAM. In J. Dinsmore, editor, *Closing the gap: symbolism vs. connectionism*, pages 113–148. Lawrence Erlbaum, Hillsdale, NJ, 1992.
10. A. Sperduti. Labeling RAAM. *Connection Science*, 6(4):429–459, 1994.
11. S. Kwasny and B. Kalman. Tail-recursive distributed representations and simple recurrent networks. *Connection Science*, 7(1):61–80, 1995.
12. S. Levy and J. Pollack. Infinite RAAM: A principled connectionist substrate for cognitive modeling. In *International Conference on Cognitive Modeling*. Lawrence Erlbaum Associates, 2001.
13. M. Adamson and R. Damper. B-RAAM: A connectionist model which de-

velops holistic internal representations of symbolic structures. *Connection Science*, 11(1):41–71, 1999.
14. T. Voegtlin and P.F. Dominey. Linear recursive distributed representations. *Neural Networks*, 18:878–895, 2005.
15. E. Oja. Neural networks, principal components, and subspaces. *International Journal on Neural Systems*, 1:61–68, 1989.
16. M. Bodén and L. Niklasson. Semantic systematicity and context in connectionist networks. *Connection Science*, 12(2):111–142, 2000.
17. D. Rohde. The simple language generator: Encoding complex languages with simple grammars. Technical Report CMU-CS-99-123, Carnegie Mellon University, Pittsburg, PA, 1999.
18. P. Li, I. Farkaš, and B. MacWhinney. Early lexical development in a self-organizing neural network. *Neural Networks*, 17(8-9):1345–1362, 2004.
19. I. Farkaš and P. Li. Modeling the development of lexicon with a growing self-organizing map. In H. Caulfield et al., editor, *Proceedings of the 6th Joint Conference on Information Sciences*, pages 553–556, Research Triangle Park, NC, 2002.
20. M. Harm. Building large scale distributed semantic feature sets with WordNet. Technical Report PDP.CNS.02.1, Carnegie Mellon University, Pittsburg, PA, 2002.
21. G.A. Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, pages 235–312, 1990.
22. R. Hadley. Systematicity of generalizations in connectionist networks. In Arbib M., editor, *The Handbook of Brain Theory and Neural Networks*, 2nd edition, pages 1151–1156. The MIT Press, 2003.
23. J. Bullinaria. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526, 2007.