Linear Recursive Distributed Representations

Thomas Voegtlin(1) and Peter F. Dominey (2)

Affiliations:

(1) Thomas Voegtlin (corresponding author),
INRIA - Campus Scientifique, B.P. 239
F-54506 Vandoeuvre-Les-Nancy Cedex, FRANCE
e-mail: voegtlin@loria.fr

(2) Peter F. Dominey
Institut des Sciences Cognitives
CNRS UMR 5015, 67 boulevard Pinel
69675 Bron cedex, France
e-mail: dominey@isc.cnrs.fr

Acknowledgments :
The authors would like to thank Prof. Paul Smolensky for his comments and suggestions. This work was partially supported by the French Ministry for Foreign Affairs, and by the Alexander von Humboldt Foundation.

Running title : Linear RAAM

Title : Linear Recursive Distributed Representations

Abstract :

Connectionist networks have been criticized for their inability to represent complex structures with systematicity. That is, while they can be trained to represent and manipulate complex objects made of several constituents, they generally fail to generalize to novel combinations of the same constituents. This paper presents a modification of Pollack's Recursive Auto-Associative Memory (RAAM), that addresses this criticism. The network uses linear units and is trained with Oja's rule, in which it generalizes PCA to tree-structured data. Learned representations may be linearly combined, in order to represent new complex structures. This results in unprecedented generalization capabilities. Capacity is orders of magnitude higher than that of a RAAM trained with back-propagation. Moreover, regularities of the training set are preserved in the new formed objects. The formation of new structures displays developmental effects similar to those observed in children when learning to generalize about the argument structure of verbs.

Keywords :

Representations, hierarchical structures, RAAM, PCA, systematicity, generalization, verb islands.

1 Introduction

In 1988, Fodor and Pylyshyn (1988) addressed a fundamental criticism to connectionism, arguing that representations in connectionist networks were inadequate to reproduce the generativity and systematicity observed in mental representations. Generativity refers to the idea that mental representations can be produced in seemingly unbounded number, generated from atomic elements according to combinatorial rules. Systematicity refers to the existence of overall symmetries, in which the ability to understand certain sentences or thoughts is intrinsically related to the ability to understand other related sentences or thoughts (e.g. a system that can represent the phrase "John loves the girl" should also be able to represent "The girl loves John"). Fodor and Pylyshyn's criticism was first based on the idea that mental representations are characterized by combinatorial syntax and semantics; the so-called 'language of thought' (Fodor, 1975). A second and equally important argument in their demonstration was that mental processes, in order to achieve systematicity, need to operate on the structure of complex representations rather than on their content. Connectionist networks, they argued, would not have the potential to represent complex structures with combinatorial properties.

In the classical point of view of Fodor, the language of thought was believed to reflect computational properties of mental processes, and the implementation level played little or no role in these processes. Although the argument by Fodor and Pylyshyn was essentially based on this denial of implementation issues, one of its merits has been to draw the attention of connectionists onto the desired properties of neural representations, and the exact nature of systematicity (Brousse, 1993; Plate, 1994; Hadley, 1994). One cause of this perceived failure of connectionism as a theory of cognitive architecture is the impossibility to use pointers and logical addresses, as in classical computer programs, in order to represent and manipulate complex structures. Shortly after Fodor and Pylyshyn's criticism, several models were proposed to address this issue (Hinton, 1990; Pollack, 1990; Smolensky, 1990). A common feature of these models is that they are connectionist, in the sense of Fodor and Pylyshyn; they are not neural implementations of a classical AI machine (for a neural implementation of a classical AI machine see Touretzky, 1990).

In general, connectionist representations are distributed. In a fully distributed representation, each concept is represented by a pattern of activities distributed over many units, and each unit is involved in representing many different concepts (Hinton et al., 1986). Hence, large numbers of objects can be represented. Some distributed representations use combinations of features, in a so-called conjunctive code; a conjunction of features is represented by the conjunction of the activity patterns that represent these features. Smolensky (1987) described how conjunctive codes can account for systematicity. In addition, in order to represent complex objects, Smolensky proposed a representation system that incorporates value/variable bindings based on tensor products (Smolensky, 1990). This system combines the advantages of feature conjunctions with the possibility of representing structured concepts using role assignment. However, representations cannot be learned in this system; they have to be predefined. Indeed, Smolensky made a distinction between connectionist theories of representation, computation and learning, and acknowledged that his theory was not a theory of learning. In addition, it is difficult to represent highly structured concepts using tensor products, because tensor products require exponentially growing networks in order to accurately represent the exponentially growing number of possible structures.

To avoid such space complexity problems for the representation of highly structured concepts, Hinton (1990) suggested that it is necessary to use time-shared resources, in order to avoid running out of hardware. He proposed to implement time-sharing by using reduced descriptions, i.e. descriptions where the representation of the same object is different depending on whether this object is seen as a part of a whole or as a whole itself. Pollack (1990) proposed a recurrent neural network that learns such reduced descriptions, the Recursive Auto-Associative Memory (RAAM), which is a recursive auto-encoder trained with error back-propagation.

The RAAM model generated much enthusiasm among connectionists, because its reduced descriptions address Fodor and Pylyshyn's challenge at a very fundamental level. Many applications were proposed. For example, the RAAM has been used to perform holistic computations, e.g. active-passive transformations of grammatical phrases (Chalmers, 1990; Chrisman, 1991), or logical transformations of symbolic expressions (Niklasson and van Gelder, 1994). Niklasson and van Gelder noted that holistic processes in RAAM are sensitive to structure, and therefore address Fodor and Pylyshyn's second argument. The RAAM was used in a model of natural language parsing, in order to represent embedded phrases (Miikkulainen, 1996). In addition, several extensions of the RAAM architecture were proposed. For example, Sperduti (1994) proposed a poweful extension that can represent arbitrary graph structures, and Kwasny and Kalman (1995) proposed a related method for encoding linked lists. More recently, Bodén and Niklasson (2000) proposed an extension of RAAM that learns representations based on context and semantics.

However, practical difficulties have been encountered with the model, which did not fully keep its promise. RAAM networks are difficult to train, and their generalization capabilities are poor (Blank et al., 1992). The scope of this paper is to present a modification of RAAM, that solves these problems. In the next section, we describe the original RAAM and we discuss its capabilities. In section 3, we propose to use linear units in a RAAM. We discuss the theoretical advantages of linear representations, and we present our model. Experimental results are presented in sections 4 and 5.

2 The classical RAAM

In this section we present the classical RAAM by Pollack (1990), and an experiment where the generalization capacity of the model is measured. We discuss the nature of generalization in this model.

2.1 The RAAM architecture

The RAAM architecture consists of two networks, an encoder and a decoder, plus some additional branching connections. The encoder produces reduced descriptions of complex structures. In a binary RAAM (see Figure 1), the encoder maps two neural populations, LEFT and RIGHT, onto a neural population WHOLE that contains a reduced description of the content of LEFT and RIGHT. Populations LEFT, RIGHT and WHOLE have the same number of neurons, k. Let vectors $\mathbf{l}(t)$ and $\mathbf{r}(t)$ denote the contents of populations LEFT and RIGHT at time t, respectively. The reduced description in WHOLE is denoted by $\mathbf{w}(t)$:

$$\mathbf{w}(t) = h(\mathbf{L}\mathbf{l}(t) + \mathbf{R}\mathbf{r}(t)) \quad (1)$$

where $\mathbf{L}$ and $\mathbf{R}$ denote the feed-forward connection matrices from LEFT to WHOLE and RIGHT to WHOLE, respectively, and $h(.)$ is a sigmoidal activation function.

Figure 1 about here

Recurrent one-to-one connections map the WHOLE population back to populations LEFT and RIGHT. Complex hierarchical structures are encoded recursively, by moving the content of the WHOLE vector to either LEFT or RIGHT, depending on the branching of the structure that is represented, while the other input population receives a new external input vector $\mathbf{x}(t)$:

$$\left\{\begin{array}{lcl}\mathbf{l}(t) & = & \mathbf{w}(t-1) \\ \mathbf{r}(t) & = & \mathbf{x}(t)\end{array}\right. \quad or \quad \left\{\begin{array}{lcl}\mathbf{l}(t) & = & \mathbf{x}(t) \\ \mathbf{r}(t) & = & \mathbf{w}(t-1)\end{array}\right.$$

After a structure has been encoded, the decoder network can be used to decode the reduced representation in WHOLE into reconstructions, in LEFT and RIGHT, using additional synapses. The atomic constituents of complex structures may be retrieved recursively, using a procedure symmetrical to the above. The whole network is an auto-encoder, and it is trained with error back-propagation.

2.2 Pollack's experiment

In order to demonstrate that his model is able to represent complex structures, Pollack (1990) trained a 20-10-20 binary RAAM on a corpus of 7 parsed phrases, which are enumerated in Table 1. These phrases were generated using the context-free grammar of Table 2. Terminals are `d,n,v,p,a`, and non-terminals are `S,NP,VP,PP,AP`. Note that due to embedding, the training set actually contains 15 distinct complex structures. We refer to this set of 15 phrases as the extended training set.

Table 1 about here

Table 2 about here

Pollack showed that his model was able to successfully encode the training set. This indeed demonstrates the validity of the concept. However, a crucial question is whether the RAAM can generalize. That is, is the network able to represent structures it has not been exposed to during its training phase? In some sense, this question is related to systematicity; if the network can be trained to represent structures made of certain constituents, and then fails to represent new combinations of the same constituents, then the representation system it defines certainly lacks a crucial feature of human representations.

In order to evaluate the generalization capability of RAAM, Pollack (1990) described an iterative procedure that identifies all the structures a network can represent, given a set of terminals. This procedure, which we will use extensively in this paper, is as follows : A generator enumerates all the possible combinations of two elements picked from a set of phrases, which initially contains only the terminals. The binary RAAM is then tested on the so formed new phrases, and phrases that are successfully encoded and decoded are added to this set. The procedure ends when the set is stable. By applying this procedure to the above network, Pollack demonstrated that it could represent the 31 additional structures of Table 3.

Table 3 about here

Hence, the encoding of new structures demonstrates that the RAAM "generalizes" to a certain extent. That is, the network uses its knowledge of certain complex structures in order to represent different structures.

2.3 Regularity of new formed structures

Regularity can be observed in the new formed structures, with a tendency to reproduce known patterns. However, grammaticality is not preserved : 12 new formed phrases of Table 3 are ungrammatical. This raises the question of the exact nature of generalization. Because grammatical phrases were used in this experiment, an obvious interpretation is to consider that the network has to perform a grammar-learning task, and that it should learn to discriminate between grammatical and ungrammatical phrases. In this interpretation, the relevant test would be, can the network figure out which sentences are ungrammatical, even though it has never been exposed to an ungrammatical sentence? In this case, the successful encoding of 12 non-grammatical structures can be perceived as a failure.

However, the original challenge addressed by RAAM was that of representing complex structures at all, using only neural activation patterns. Considering this challenge, it is certainly more appropriate to consider generalization as the capacity to represent any unknown complex structure, whether grammatical or not. Although representing any structure is certainly not an interesting accomplishment for a non-connectionist model - a symbolic system is able to represent all trees -, it is a relevant challenge for a connectionist system. Considering this challenge, the encoding of 31 unknown phrases is a valuable accomplishment.

This does not mean that the question whether generalization follows a set of rules is uninteresting. In fact, the characterization of new formed structures is a crucial point. However, an interpretation based on grammaticality would probably fail to answer this question; there seems to be little reason why the simple architecture of RAAM would be sufficient to learn the rules of a context-free grammar. Moreover, such an interpretation would be restricted to phrases, whereas the range of application of RAAM is much broader.

2.4 Capacity

A second observation that can be made about the set of new formed structure is that it is relatively small. There exists a near-infinite number of ways of possibly combining the terminals of the training set, in comparison to which a system than can represent only 31 unknown structures seems limited. In fact, poor generalization has been identified as one of the most important shortcomings of RAAM (Blank et al., 1992; Plate, 1994), because it imposes a severe limitation on the systematicity of its representations. In order to represent structurally new examples, a RAAM typically needs additional training, which might wipe out existing representations.

In contrast, the ability to represent structurally new concepts is a major component of human mental representations. For example, it accounts for the diversity of constructions in human language (whether syntactically well-formed or not), or in other cognitive activities. Traditionally, this form of generalization has been considered to result from the use of generative rules, hence the term "generativity". However, the generalization ability of most neural networks does not result from the use of generative rules; it rather results from the possibility to combine a set of learned features in a distributed representation. For example, the hidden units of a multilayer perceptron detect features of the input that are relevant for predicting the output (Rumelhart et al., 1986); correct generalization occurs due to this relevance, by using new combinations of features.

3 Linear RAAM

In this section we argue that the only modest capacity of RAAM is not inherent to its concept, but is rather an artifact of the learning method employed. We propose to use a RAAM with linear neurons, and to train it with Oja's learning rule rule (Oja, 1989).

3.1 Advantages of using linear units

At first, using linear instead of non-linear units might appear as a restriction. This is because a one hidden-layer network of sigmoidal units trained with error back-propagation is a so-called universal approximator. That is, it is able to learn any complex mapping between its inputs and its outputs, with a level of precision that only depends on how many hidden units are used. In contrast, the set of input-output mappings that can be performed by a linear network is more limited.

However, the RAAM is a recursive auto-encoder. This means that the mapping to be learned between inputs and outputs is far from complex - it is the identity function - and that it is not possible to rely on a high number of hidden units - the goal here is precisely to build a reduced representation. Hence, a RAAM based on linear units is not a restricted form of RAAM.

A second argument in favor of linear units comes from the desired properties of a representation system. We have seen that representations should have some form of systematicity, and that systematicity in distributed representations can be obtained by using a conjunctive code, where conjunctions of neural activity patterns encode conjunctions of features. We shall demonstrate here that linearity allows a RAAM to achieve a conjunctive code. Moreover, if conjunctions of continuous neural activity patterns are implemented additively, then linearity is necessary for a conjunctive code.

In order to demonstrate this, let us consider that a RAAM is trained to represent a set of complex structures, and let a and b denote two possible features of these complex structures. These features may refer to the content or to the structure of represented objects. Let $R(a)$ and $R(b)$ denote the non-reduced neural representations of a and b respectively. For example, if the RAAM is binary, $R(a)$ is an activity pattern distributed over both populations LEFT and RIGHT. The encoder transforms the non-reduced representation into a reduced representation in WHOLE. Hence, let $r(a)$ and $r(b)$ denote the reduced representations of $R(a)$ and $R(b)$ respectively.

A conjunctive code is a code such that $R(a) \wedge R(b) = R(a \wedge b)$, where $\wedge$ denotes the conjunction. To define this condition properly, we need to specify how conjunctions of neural activities are implemented. With binary neurons, an activity equal to 1 classically means that the unit is firing, and a conjunction of two firing patterns is implemented by a logical OR of the corresponding vectors. However, if neural activities take on continuous values, it is necessary to extend this definition. A classical way to do this is to implement a conjunction with a sum (Smolensky, 1990; Plate, 1994). This choice is consistent with the classical interpretation of continuous activities in terms of firing rates. This yields:

$$R(a \wedge b) = R(a) + R(b) \tag{2}$$

Let f denote the activity function of the encoder, which transforms a non-reduced representation into a reduced representation (for convenience, we incorporate both effects of the synaptic weights and of the transfer function into the notation f). We may write:

$$r(a) = f(R(a)) \tag{3}$$

$$r(b) = f(R(b)) \tag{4}$$

$$r(a \wedge b) = f(R(a \wedge b)) \tag{5}$$

Since reduced representations may be used as a part-of-whole through recurrent connections, the neural code must have the same properties in the reduced and non-reduced representations. Hence, equation (2) should hold for reduced representations too:

$$r(a) + r(b) = r(a \wedge b) \tag{6}$$

From the above equalities we can derive :

$$f(R(a)) + f(R(b)) = f(R(a) + R(b)) \tag{7}$$

The only assumption we have made so far on vectors $R(a)$ and $R(b)$ is that they correspond to encoded features. However, activity patterns might be imposed on the RAAM, for example if they encode terminals. Since we do not want a network that requires constraints on the encoding of terminals, the above linearity relation must hold for any vectors $R(a)$ and $R(b)$. This imposes that f must be linear.

This does not mean that a RAAM based on nonlinear units will not work, or that it will not be able to represent objects by conjunctions of features. However, it is convenient to implement conjunctions of features by sums of vectors; in this case, we have demonstrated that a RAAM must use linear units in order to be compatible with a conjunctive code.

3.2 Description of the network

Here we describe the implementation of a linear RAAM. We use the same notations as in the previous section. The activity, $\mathbf{w}_i(t)$, of neuron i in WHOLE reads :

$$\mathbf{w}_i(t) = \sum_{j=1}^{k} \mathbf{L}_{ij}\mathbf{l}_j(t) + \sum_{j=1}^{k} \mathbf{R}_{ij}\mathbf{r}_j(t) \tag{8}$$

where $\mathbf{l}_j$ is the activity of neuron j in LEFT, $\mathbf{r}_j$ is the activity of neuron j in RIGHT, $\mathbf{w}_i$ is the activity of neuron i in WHOLE, $\mathbf{L}_{ij}$ is the weight from a unit j in LEFT to unit i in WHOLE, and $\mathbf{R}_{ij}$ is the weight between a unit j in RIGHT to unit i in WHOLE. In matrix form, this reads :

$$\mathbf{w}(t) = \mathbf{L}\mathbf{l}(t) + \mathbf{R}\mathbf{r}(t) \tag{9}$$

where the weight matrices of the encoder, corresponding to the connections from LEFT to WHOLE and RIGHT to WHOLE, are denoted by $\mathbf{L}$ and $\mathbf{R}$, respectively. Before a new structure is stored in the linear RAAM, all neural activities are reset to zero.

In a linear auto-encoder, it is possible to use the transpose of the encoder matrix for the decoder, with no loss of degree of freedom (Baldi and Hornik, 1988). Let $\mathbf{L}'$ and $\mathbf{R}'$ denote the transposes of $\mathbf{L}$ and $\mathbf{R}$, respectively. By applying $\mathbf{L}'$ and $\mathbf{R}'$ to $\mathbf{w}(t)$, we generate the reconstructions $\bar{\mathbf{l}}(t)$ and $\bar{\mathbf{r}}(t)$ of $\mathbf{l}(t)$ and $\mathbf{r}(t)$, respectively :

$$\begin{cases} \bar{\mathbf{l}}(t) = \mathbf{L}'\mathbf{w}(t) \\ \bar{\mathbf{r}}(t) = \mathbf{R}'\mathbf{w}(t) \end{cases}$$

Minimizing the mean-square error between encoded vectors and the output of the decoder results in the following stochastic learning rule :

$$\forall 1 \leq i \leq k, \forall 1 \leq j \leq k, \begin{cases} \Delta\mathbf{L}_{ij} = \eta\mathbf{w}_i(t)\left(\mathbf{l}_j(t) - \sum_{l=1}^{k}\mathbf{L}_{lj}\mathbf{w}_l(t)\right) \\ \Delta\mathbf{R}_{ij} = \eta\mathbf{w}_i(t)\left(\mathbf{r}_j(t) - \sum_{l=1}^{k}\mathbf{R}_{lj}\mathbf{w}_l(t)\right) \end{cases}$$

where η is a learning rate. Note that this corresponds to Oja's constrained Hebbian learning rule (Oja, 1989) applied to vectors $\mathbf{l}$ and $\mathbf{r}$ simultaneously. Hence, the learning rule will attempt to extract the principal components of the joint distribution of vectors $\mathbf{l}(t)$ and $\mathbf{r}(t)$.

Principal Components Analysis (PCA) consists of finding a set of orthogonal basis vectors that correspond to the directions of highest variance of the distribution of an input vector. PCA is known to provide optimal linear auto-encoding with respect to mean-squared error. However, it should be emphasized that the operation of our model is far more complex than PCA. Vector $[\mathbf{l}(t); \mathbf{r}(t)]$ contains $\mathbf{w}(t-1)$, which results from recurrent connections. Therefore the distribution of vector $[\mathbf{l}; \mathbf{r}]$ is not defined a priori, and it results from the internal representations devised by the network. Hence, learning is a so-called "moving target" problem[Note1].

The above algorithm generalizes PCA to tree-structured recursive data. A tail-recursive version of this algorithm has been presented in (Voegtlin, 2000; Voegtlin and Dominey, 2001), for the representation of temporal context. The tail-recursive version generalizes PCA to time series, and it differs from classical linear methods used in time-series analysis, such as auto-regressive models (Bryson and Ho, 1975; Soderstrom and Stoica, 1989), in that it is not based on a prediction.

Only spatially and temporally local terms are involved in Oja's learning rule, which makes it compatible with basic biological constraints, and more plausible than error back-propagation. Moreover, symmetrical weights are used for feed-forward and feedback connections. From a biological perspective, this means that the transposes of $\mathbf{L}$ and $\mathbf{R}$ can be implemented in a separate set of synapses, trained with a similar local learning rule.

3.3 Terminal test

For the decoder of RAAM, a "terminal test" is necessary, in order to decide whether a decoded vector encodes a terminal element or a non-terminal. If a decoded vector (LEFT or RIGHT) is a non-terminal, then further decoding is necessary. However, encoding is in general not perfect; a vector might be altered in the encoding-decoding process. This might lead to failures of the termination test, and problems known as "infinite loops"or "terminating non-terminals". A classical termination test assigns a Boolean "terminal code" to vectors, adding a dedicated unit to the network (Stolcke and Wu, 1992).

In this paper, the terminal test we used for both the linear RAAM and the classical RAAM simply consisted in measuring the Euclidean distance between decoded vectors and vectors encoding terminals. 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 threshold, θ. Using the Euclidean distance is consistent with the distance considered in error minimization. However, it is slightly different from the test in Pollack (1990), which involved a non-Euclidean distance. This difference is not significant for the results we present.

In decoding a complex structure, whenever an infinite loop, a terminating non-terminal or a wrong terminal is encountered, we consider that encoding is unsuccessful. However, the termination test in itself is not sufficient to ensure that the answer of the decoder is correct; incorrect answers might occur if the test is too permissive, that is, if θ is too high. Thus, in order to assess performance of our network it is necessary to check that the decoded structure is the correct one. We performed this additional test in all our experiments.

3.4 Related model

The idea of using linear units in a RAAM was first proposed by Callan and Palmer-Brown (1997) . In order to generate the encoding matrix, these authors proposed an algorithm based on Principal Components Analysis (PCA). They reported fast training, and improved holistic processing. However, their generation algorithm departs from connectionnist principles, which are the essential motivation of the RAAM model. In addition, the size of the representations resulting from this algorithm depends on the training set, and it cannot be fixed in advance. This is because PCA is used to remove only the directions of null variance, while all other components of the distribution are retained. As a result, no compression is performed : the size of the reduced representation is the same as the dimension of the space spanned by the input to the encoder.

In contrast, the model presented above only slightly deviates from the original RAAM. Like in the original model, the size of the representation does not depend on the data. Hence, the model may be trained on datasets of dimension higher than the size of the reduced representation. In this case, the reduced descriptions actually have reduced dimension, in the sense that some components of the distribution (those corresponding to the directions of smaller variance) are removed by the encoder. Hence, the compression performed by the encoder is lossy, in the sense that some information is lost when smaller components are removed. However, the encoded structures can still be recovered, because the terminal test reintroduces the information that has been lost during encoding. In what follows, we will show that, due to this dimension reduction, our model displays unprecedented generalization capacities.

4 Performance study

In this section we present the results of computer simulations. These simulations consisted in a systematic comparison of the generalization performance of a RAAM trained with back-propagation, and a linear RAAM trained with Oja's rule.

4.1 Reproducing Pollack's experiment

We reproduced the experiment by Pollack (1990) described above, using a linear RAAM and classical RAAM trained with back-propagation. Both networks were of size 20-10-20, and they were trained on the 7 phrases of Table 1. The encoding of terminals was the same as in the original Pollack experiment : each of the five terminals was coded by a 1-in-5 bit pattern, padded with five zeros for the five remaining neurons^{Note2}. Note that although the 5 neurons of the input are not used in the representation of terminals, they are crucial to the overall capacity of the network, because they are used in the representation of complex structures. The initial synaptic weights were picked at random in the interval $[-0.5; 0.5]$ with uniform distribution. Both networks used exactly the same termination test, as described above.

The linear RAAM was trained for 3000 iterations with learning rate equal to 0.01. The classical RAAM was trained for 300 iterations only, with the same learning rate. This difference in training time is made in order to maximize the performance of the classical RAAM, which decreases when training is pursued for more than about 300 iterations. The effect of training time will be investigated in more detail later.

After training, the complete set of encodable structures was enumerated using the comprehensive procedure described above. Values of θ were tested in the interval $[0; 1.6]$. Since the Euclidean distance between terminals is $\sqrt{2}$, we do not expect successful reconstructions to occur for $\theta > 1.41$, unless by pure chance. The number of successfully encoded phrases is plotted in Figure 2, using a logarithmic scale for the vertical axis.

Figure 2 about here

Figure 2 shows that the performance of both models is sensitive to the threshold θ. There exists an optimal value of θ, which is around 0.90 for the linear RAAM, and around 1.05 for the classical RAAM. The existence of this optimum is easy to explain. If the terminal test of the decoder is too permissive, decoding errors will occur. Alternatively, if this test is too restrictive, fewer phrases will be reconstructed.

We observe a large difference in the number of phrases that can be represented by the two models. The maximal number of phrases successfully represented by the classical RAAM was 122, for $\theta = 1.05$, which is of the same order of magnitude as the 15+31 successfully encoded phrases reported by Pollack (1990). In contrast, the linear RAAM was able to represent 1989 phrases with $\theta = 0.90$. Hence, we observe a dramatic improvement of capacity.

4.2 Effect of over-training

In the previous experiment the classical RAAM was trained for less iterations than the linear RAAM. This is because the back-propagation procedure is sensitive to over-training. That is, generalization performance decreases when learning is pursued beyond a certain stage. This occurs because the examples of the training set become over-fitted, at the expenses of the relevance of the input-output mapping.

We investigated the influence of training time on both linear RAAM and classical RAAM. We trained the networks described above on Pollack's corpus. We mentioned earlier that the optimal value of the reconstruction threshold, θ, is around 0.9 for linear RAAM, and around 1.05 for classical RAAM. These values are independent of the number of training iterations, and were used here.

We measured the number of phrases that can be represented, for numbers of training iterations between zero and 10^6. Results are reported in Figure 3, using a logarithmic scale for both time and performance. Performance is averaged over 25 random initial conditions, and error bars indicate the standard deviation.

Figure 3 about here

The average performance of the classical RAAM is maximal for training times around 300 iterations, after which it decreases. Eventually, the number of encoded phrases goes back to 15, which means that the ability to represent new phrases entirely vanishes.

In contrast, no over-training effect is observed with the linear RAAM. The mean performance monotonically increases, and stabilizes around 2000 phrases. Note that average performance does not reflect individual experiments; performance during individual experiments may be non-monotonic (data not shown). Standard deviation is higher for linear RAAM than for classical RAAM, which denotes a higher sensitivity to initial conditions.

Figure 4 about here

Over-training typically occurs because error minimization generates an input-output mapping that matches the training points too well, but that has an irrelevant shape for other points. Figure 4 indicates the evolution of the mean reconstruction error performed on the structures of the training set during learning, in the above experiments. We observe that the average error of the linear RAAM quickly reaches a stable value. In contrast, the average error of the RAAM trained with back-propagation decreases throughout the experiment. This reflects a crucial difference between the two learning methods. PCA tends to result in a stable reconstruction error, that corresponds to the directions of smaller variance of the $[\mathbf{l}; \mathbf{r}]$ vector. In contrast, back-propagation minimizes the error at the cost of the mapping.

4.3 Influence of the training set

In general, the generalization performance of a neural network crucially depends on the number and variability of examples in its training set. In the preceding experiment, 7 training examples were used, and the representation used 10 neurons. It would be interesting to know if a representation with 10 neurons can handle more training examples, and how generalization is affected by this. We trained the same networks as above on the training set of Table 4, which contains 14 phrases.

Table 4 about here

All these phrases are of size 6. The same training and generalization procedures as above were used. The numbers of training iterations were the same as in the above experiment. Generalization performance is reported in Figure 5.

Figure 5 about here

Figure 5 shows that the performance of the classical RAAM is not greatly improved by increasing the size of the training set; the maximal number of phrases successfully decoded is 160, for $\theta = 1.10$. This suggests that the size of the training corpus chosen by Pollack was more or less optimal given the size of the network. In contrast, the performance of the linear RAAM is dramatically improved by the addition of new elements to its training set; 15988 phrases are successfully represented for $\theta = 0.85$, which is more than 1000 times the size of the training set, and about 100 times the capacity obtained with back-propagation. Analyzing the set of successfully encoded phrases reveals that most new formed phrases have more levels of embedding than the examples of the training set, and that they may be much longer. All the phrases of the training set were of length 6, and had depth below or equal to 5. In contrast, the set of phrases successfully encoded with $\theta = 0.85$ contained phrases of length up to 46, and of depth up to 10. A successfully encoded phrase of length 46 was:

```
((((a(d(d(an))))((pv)((vv)((a(p(dv)))(p(dn))))))(p(d(an))))(((a(pv))n)
(((a(d(d(an))))(ad))((an)(((a(d(d(an))))(ad))((an)((dv)(pv))))))))
```

The experiments presented in this section demonstrate that the generalization capacity of RAAM is dramatically improved by using linear neurons instead of non-linear neurons, and Oja's learning rule instead of error back-propagation. Another important observation is that generalization capacity is greatly augmented when the training set contains more examples. This addresses major criticisms that have been made to the RAAM model. In the following sections, we will investigate the mechanism of generalization in the linear RAAM.

5 Interpretation: generalization as linear combinations

In the previous section, we observed a dramatic improvement of the generalization capacity of the linear RAAM after new elements were added to the training set. One possible explanation is that the network formed new representations by combining learned representations. This hypothesis is investigated here.

In order to do this, we consider the initial corpus of Table 1. We have shown that a 20-10-20 linear RAAM is able to learn these phrases. However, Figure 2 shows that no phrase is reconstructed when the error threshold θ is close to zero. In fact, there exists a minimal value of θ, below which the corpus cannot be decoded. For a representation using $k = 10$ neurons, the minimal value of θ is 0.34.

Table 5 displays the minimal value of θ for different sizes of the representation. In all these experiments the encoding of terminals was using 5 bits; each terminal was encoded with one bit to one and the four other bits to zero. The remaining units were not used in the representations of terminals, and they were set to zero. At least 7 units are necessary to perform the task.

Table 5 about here

Interestingly, for $k \geq 12$ neurons, θ can be chosen arbitrarily close to zero. For this, the learning rate η needs to be of more or less the same order of magnitude as the threshold^{Note3}. For example, a linear RAAM with 12 neurons would learn the corpus with precision $\theta < 0.00001$, using a learning rate $\eta = 0.0001$.

Hence, the extended corpus can be encoded and decoded with no reconstruction error using only 12 neurons, while this corpus contains 15 different phrases^{Note4}. This can be interpreted in terms of linear dependencies. The fact that 15 different objects can be represented with high precision using only 12 neurons suggests that there are 3 linear dependencies between the vectors encoding the elements of the extended corpus.

In order to test this hypothesis, we computed the rank of matrices formed with the vectors encoding the elements of the extended corpus. Let P denote the 12 by 15 matrix whose columns are vectors of activities in WHOLE, encoding the phrases of the extended corpus. P is of rank 12, and rank computations demonstrated that columns 9, 11 and 12 are linear combinations of other columns with indexes strictly smaller than their own index. The phrases corresponding to these columns are :

```
(p(d(an)))

(a(an))

(a(a(an)))
```

If we remove columns 9,11 and 12 from matrix P, we obtain a 12 by 12 full-rank matrix, denoted by Q. Q is a change of basis matrix in the space of the reduced representations, from the canonical basis into a more intuitive basis made of free vectors that encode the elements of the extended corpus.

We computed the inverse Q^{-1} of Q, and the product $Q^{-1} \times P$. To precision 10^{-6}, the result was the following matrix :

$$Q^{-1} \times P = \begin{bmatrix} +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & +1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \end{bmatrix}$$

Interestingly, columns 9, 11 and 12 have integer coordinates. This means that the vectors representing the phrases corresponding to these columns are discrete linear combinations of the vectors representing the other phrases. The values of the other columns could have been predicted a priori. From the coordinates of columns 9, 11 and 12, we see that the linear relations between the distributed representations of the elements of the extended corpus are :

```
(p(d(an)))  =  - (v(dn)) + (v(d(an))) + (p(dn))
(a(an))     =  - (dn)    + (d(an))    + (an)
(a(a(an)))  =  - (dn)    + (d(an))    - (v(dn)) + (v(d(an))) + (an)
```

In the above expressions, we have written linear combinations of nested structures. This refers to the linear relations between the internal representations devised by the RAAM. However, we can define 'plus' and 'minus' operators on binary trees in a way that is consistent with the function of linear RAAM. Consider the following equality :

```
(p(d(an))) = - (v(dn)) + (v(d(an))) + (p(dn))
```

Figure 6 about here

In order to add or subtract phrases, we superimpose the corresponding labeled binary trees, and we add or subtract labels. For example, figure 6 shows the trees associated with phrases `(v(dn))`, `(v(d(an)))` and `(p(dn))`. A new tree, corresponding to the addition of `(v(d(an)))` and `(p(dn))`, is built by superimposing the trees that correspond to these phrases, where each node or leaf receives a label equal to the sum of the labels of the leafs of `(v(d(an)))` and `(p(dn))` that are at the same position. If there is no label at a node, a label equal to zero is assumed. The label of a leaf is equal to the corresponding terminal. Note that the tree that we have formed with `(v(d(an)))` and `(p(dn))` does not represent a nested phrase; some labels are placed at internal nodes, and there are leafs with labels `2d` and `v+p`, which are not terminals. However, if we subtract the tree `(v(dn))` from the tree `(v(d(an))) + (p(dn))`, we obtain a valid tree (a tree that corresponds to a phrase); only leafs have nonzero labels, and all labels are terminals.

We have defined addition and subtraction on nested structures. These operators are not internal (i.e. they do not always result in a valid description of a nested structure). However, certain examples of the training set are linear combinations of others. We have demonstrated that these linear relations are preserved in their neural representations. Hence, linear dependencies within the training set are exploited by our model.

In order to test if linear combinations of learned structures can account for generalization, we applied the enumeration procedure to the 24-12-24 RAAM, using $\theta = 0.00001$. This resulted in a total of 34 encodable structures, of which 19 are new (see Table 6).

Table 6 about here

The 19 new formed structures of Table 6 are much less than what we obtained with higher values of θ. However, we used $\theta \approx 0$ here in order to obtain a high level of precision. By multiplying the inverse of matrix Q with the vectors representing these new formed phrases, we checked that each of these phrases was a linear combination of phrases from the extended training set. For instance:

```
((dn)(p(d(an)))) = -(d(an))+(d(a(a(an))))+((dn)(p(dn)))
                    +(v(dn))-(v(d(an)))
(v(d(a(an))))    = -(d(an))+(d(a(a(an))))+(v(dn))
(p(v(dn)))       =  (d(an))-(d(a(a(an))))-((dn)(p(dn)))
                    -2*(v(dn))+(v(p(dn)))+(v(d(an)))
                    +(p(dn))+((dn)(v(d(an))))
```

This demonstrates that all the new formed phrases are valid linear combinations of learned examples, and all valid linear combinations of the learned examples can in turn be represented. Hence, generalization obeys the rules of linear algebra. As a consequence, a training set can be reduced to a small number of well-chosen (i.e. linearly independent) examples.

New formed phrases are formed by valid linear combinations of binary trees, which may be considered as a set of symbolic rules. However, it should be emphasized that the network does not "follow" these rules, in the sense that a symbolic rule system follows rules while performing an inference. The only rules that are "followed" by the network are those that describe how activities are updated, depending on previous activities. Hence, our model is fully connectionist, and it is not a neural implementation of a symbolic rule system. Linear combinations only describe what new structures can be represented by the model.

5.1 Considerations on generalization

The analysis of generalization proposed in the previous paragraph holds only if the reconstruction threshold, θ, is close to zero. However, we observed that capacity is orders of magnitude higher when θ is significantly different from zero. In addition, linear combinations cannot result in structures that are deeper than their constituents, like the structures we observed in section 4. This suggests that linear combinations are not sufficient to account for generalization.

In fact, increasing the error threshold offers the possibility of representing objects imperfectly, by removing the directions of smaller variance. The information that is lost during this encoding is recovered when the terminal test is performed. Imperfectly represented objects might be much deeper than the learned examples, as observed in the above experiments. Moreover, the drastic improvement of capacity between figures 2 and 5 suggests that imperfectly represented structures can also be linearly combined.

An important question is whether valid linear combinations of learned items can always be represented. For $\theta = 0$, we have seen that it is the case. In the general case ($\theta \neq 0$), we need to consider more precisely the role of reconstruction errors. Imperfectly represented structures may be linearly combined. In this case, the norm of the error associated with a linear combination of imperfectly represented structures will be below or equal to the sum of the norms of reconstruction errors associated with its constituents. Hence, it will be correctly decoded if this sum is below θ. This will depend on the particular structures that are encoded. This point will be investigated in more detail in the next section.

6 Consistent generalization

So far we have demonstrated that a linear RAAM, trained on a small set of examples, can generalize to large numbers of complex structures. We have observed that new formed structures may be longer and deeper than the training examples. Performance was measured by the number of structures that could be correctly represented. In doing this, we used grammatical phrases and we observed that most new formed phrases were not grammatical. We do not consider this as a failure, because the original challenge was to be able to represent large sets of unknown structures at all. However, the question whether new formed structures follow a rule is nonetheless very important.

Although the previous section gave a good insight on how new structures are formed, this characterization in terms of linear combinations is quite abstract, and does not indicate which regularities our system is capable of learning. It would be more interesting to find a characterization in terms of consistency. Consistency can be defined with respect to rules followed by the examples of the training set, that are then preserved in the new formed examples. We have observed that grammaticality is not preserved in the new formed examples. This does not mean that consistency cannot be found in the way our model generalizes, though; we might have to look for consistency at a different level.

6.1 The propositional experiment

As an illustration of the systematicity of mental representations, Fodor and Pylyshyn considered the symmetry of certain types of concepts :

"What does it mean to say that thought is systematic? Well, just as you don't find people who can understand the sentence 'John loves the girl' but not the sentence 'the girl loves John', so too you don't find people who can think the thought that John loves the girl but can't think the thought that the girl loves John."
(Fodor and Pylyshyn, 1988)

Pollack (1990) demonstrated that a RAAM trained on a set of 13 semantic propositions was able to generalize in this way. He used the following propositions in this experiment :

1. Pat loved Mary.
2. John loved Pat.
3. John saw a man on the hill with a telescope.
4. Mary ate spaghetti with chopsticks.
5. Mary ate spaghetti with meat.
6. Pat ate meat.
7. Pat knew John loved Mary.
8. Pat thought John knew Mary loved John.
9. Pat hoped John thought Mary ate spaghetti.

10. John hit the man with a long telescope.

11. Pat hoped the man with a telescope saw her.

12. Pat hit the man who thought Mary loved John.

13. The short man who thought he saw John saw Pat.

A ternary representation of the above sentences is devised in Table 7. This representation encodes propositions in a recursive (ACTION AGENT OBJECT) order, where "`mod`" is used to specify adjectives in triples, and "`is`" serves as subject-raiser. Terminals belong to one of the five following classes : THING, HUMAN, PREPOSITION, ADJECTIVE, and VERB. Note that the ambiguous sentence number 10 has been translated into two different representations.

Table 7 about here

Pollack (1990) demonstrated that a 48-16-48 ternary RAAM, trained on the 14 structures of Table 7, was able to represent all the possible combinations of `(loved x y)`, where `x` and `y` belong to the set {`John Mary Pat man`}, thus displaying some degree of systematicity. Generalization was semantically consistent, because `Mary, Pat, John` and `man` all belong to the class HUMAN, which makes all `(loved x y)` combinations meaningful[Note5]. A crucial question is, to what extent does this consistency result from learning? It is important to assess the role of learning, because it will allow us to understand what type of inference can be performed in a RAAM.

In order to answer this question, it is necessary to distinguish between internal and external regularities of the training set. Internal regularities can be learned, and contribute to successful generalization through inference. For example, the RAAM has been exposed to propositions where `Mary`, `John` and `Pat` play similar roles (e.g. they are AGENTs or OBJECTs of the same ACTION, `loved`). If inference is performed based on the common properties of `Mary`, `John` and `Pat`, then it will favor meaningful generalization. We want to study the effect of this type of internal regularity.

In contrast, external properties will not be learned, but may influence generalization. For example, we will see that terminal encoding may favor meaningful generalization. Another external property of propositions is semantics. Since propositions of the training set have no external referent, semantics are external to the training set. However, learnable regularities will necessarily be restricted to internal regularities of the training set. Hence, human-like semantics preservation is not achievable. Therefore we will not define consistency in terms of human semantics. For example, it is considered that `with` and `mod` are ACTIONs in the training set. This will have consequences on how the network generalizes.

6.2 Internal properties of the training set

The most obvious internal regularity of the training set is word order. Terminals are used at specific positions. For example, `loved` always occurs in the ACTION field of a triple. More precisely, the word-order regularities of the training set are the following : elements of the class VERB mainly occur in the ACTION field, and sometimes in the AGENT field; elements of the class ADJECTIVE always occur in the OBJECT field; PREPOSITIONs are always in the AGENT field; HUMAN and THING never occur in the VERB field. We may look for consistency at this level in the new formed propositions. Note that if generalization preserves word order, then new-formed propositions will have a natural tendency to be meaningful to humans. This is only a tendency, however, because the relation between word-order and semantics is not very strict. For example, `(loved ate chopsticks)` is consistent with word order, because the training set contains a proposition where `ate` is in the AGENT field[Note6]. However, `meat` never occurs as AGENT. Hence, `(loved meat John)` is not consistent with word order, while `(loved John meat)` is.

Beyond word order, the training set has other properties. For example, the AGENTs and OBJECTs used with `loved` always belong to the set {`John Mary Pat man`}. Hence, the co-occurrence of certain ACTIONs with certain AGENTs and OBJECTs is an internal regularity of the training set. The proposition `(loved man John)`, although consistent with word-order, violates this regularity, because `man` never occurs as AGENT of the ACTION `loved`. In contrast, `(loved Mary Pat)` is a new formed proposition that is consistent with this regularity. We call this type of regularity context-specificity. Context-specificity is a stronger constraint than word order. Context-specificity and word order define two different levels of consistency, for which we evaluated our network.

6.3 Effects of terminal encoding on generalization

In his propositional experiment, Pollack used the 16-bit terminal encoding of Table 8. This code is designed in a way that facilitates meaningful generalization. This is because the membership of words in classes such as THING, HUMAN, PREPOSITION, ADJECTIVE, and VERB, is explicitly indicated by dedicated bits. Hence, vectors that belong to the same class will be topologically closer than vectors from different classes. Semantically consistent generalization might largely result from such a choice (Hadley, 1994).

Table 8 about here

It may be useful to favor meaningful generalization by choosing a well-suited terminal encoding. For example, Bodén and Niklasson (2000) proposed an extension of RAAM where the representations of terminals are *learned*, in order to achieve semantic systematicity. In this extension, the representations of terminals are learned by a separate encoder network, based on the internal regularities of the training set. However, our aim here is to investigate the generalization properties of the linear RAAM model, and more precisely its ability to discover structural regularities in its training set, that is, structural relations between its constituents. We may do this by demonstrating that these regularities are preserved in the new formed structures. In doing this, we cannot afford to use a well-suited terminal encoding, which would constitute a bias, because the properties of terminal encoding are external properties, not structural regularities. If it is correlated with structural relations, terminal encoding might provide additional information about word classes, which is the case in Table 8. In order to test generalization, it is necessary to eliminate this influence.

In order to preclude any influence of terminal encoding, we designed a neutral code, where vectors contain no information about the type of terminal they represent. This code uses 32 neurons, and each of the 23 vectors encoding a terminal has only one bit equal to 1 and all its others bits are zero. Hence, the Euclidean distance between two vectors encoding terminals is always $\sqrt{2}$. This code is neutral because all vectors are equivalent and no *a priori* class information is available.

In order to test the influence of terminal encoding, we trained a 96-32-96 ternary linear RAAM on the above propositions, using this neutral code. After training, we observed that the linear RAAM was able to represent 100% of all `( loved x y )` combinations, with `x` and `y` in the set { `John Mary Pat man` } , and

$\theta = 0.7$. In comparison, a 96-32-96 classical RAAM trained in the same conditions would only achieve 50% of successful encoding on these phrases ($\theta = 0.7$). If the terminal encoding of Table 8 was used, both linear and classical RAAM would successfully represent 100% of these combinations. This demonstrates that the encoding of Table 8 favors meaningful generalization in the classical RAAM trained with back-propagation. However, the same bias can be observed with the linear RAAM as well; for example, if the error threshold was $\theta = 0.4$ then the linear RAAM using the neutral code would represent only 75% of the `( loved x y )` combinations, versus 100% with the encoding of Table 8. For this reason, we used the neutral code in what follows.

6.4 Generalization with two verbs

In order to investigate consistency, we considered all the triples that can be formed with the verbs `loved` and `ate`, and the agents and objects associated with these verbs. We enumerated the combinations that can be formed at the different levels of consistency defined above : word order and context-specificity. Then we examined the generalization performance of linear RAAM on these items.

The verbs `loved` and `ate` occur with no embedding in 6 propositions of the training set, which are enumerated in the first list of Table 9 ("Training set").

Table 9 about here

It is possible to recombine the words from these propositions, in a way that is consistent with both word order and context-specificity. This results in 7 new triples, that are enumerated in the second list of Table 9. Due to context-specificity, nouns of this list occur with the same verb as in the training examples. For this reason, this list is called "within-verb generalization".

It is also possible to recombine these words in a way that is consistent with word-order, but that violates context-specificity. This results in the 26 triples of the "between-verb generalization" list of Table 9. In the last nine triples of this list, `spaghetti` is AGENT, which is rather counter-intuitive. However, this is consistent with the regularities the network is exposed to; word-order is preserved because `( with spaghetti meat )` occurs in the training examples.

Finally, we may consider combinations of these words that violate word-order. We restrict the enumeration to propositions where the verbs `loved` and `ate` are in the ACTION field. This results in the last list of Table 9, "Word order violations". These propositions violate word-order because the word `meat` never occurs in the AGENT field of a triple in the training set.

6.5 Verb islands

We measured the success of linear RAAM at encoding the propositions of Table 9. The percentage of correct encoding over time is reported on Figure 7 for each list of Table 9. Data was averaged over 30 experiments whith random initial conditions, and the reconstruction threshold was $\theta = 0.7$.

Figure 7 about here

We observed that word-order violations never occur. In addition, different learning dynamics were observed on the above sets of propositions.

- The examples from the training set were learned first, after about 5 presentations of the training set.
- The second set of propositions, where word-specificity is respected ("within-verb generalization") could be represented later, after about 15 presentations of the training set.
- The learning curve for propositions that violate word-specificity ("between-verb generalization") is much slower. Performance reached 80% after 50 presentations of the training set.

Hence, generalization preserves word-order. In the early stages of training, generalization is context-specific. Later, the network learns to use words in different contexts than their context of apparition in the training examples.

This early preference for context-specific generalization closely resembles the acquisition of syntactic competence in young children. Tomasello (1992) observed that children during the period from 15 to 24 months learn to use verbs in a very uneven manner. Most verbs used by children at this age are used in only one or two verb-argument configurations. Moreover, children seem to have very few linguistic abstractions, namely the categorization of nominals used in conjunction with certain verbs. These observations are summarized in the "Verb Island Hypothesis" (Tomasello, 1992), which states that childrens' early language is entirely organized and structured around individual verbs, which have open nominal slots. A consequence is that the categories that are made by children are verb-centered :

"This means that syntactic categories with which children are working are not such verb-general things as 'subject' and 'object', or even 'agent' and 'patient', but rather such verb-specific things as 'hitter' and 'hittee', 'sitter' and 'thing sat upon' "(Tomasello, 2000)

One implication of these observations is that young children do not have the same syntactic competence as adults. Tomasello's theory of language acquisition is usage-based, and it is opposed to the use of adult-like grammars to describe young children's language. It also challenges the idea that human representations are always systematic.

The progressive use of words in novel contexts is observed in the way our model generalizes. In the new formed phrases, nouns initially occur with the same verbs as in the training examples. For example, constructions based on the verb `ate` preferentially involve agents and objects that have been used with this verb ('eaters' and 'edible things'). Hence, the effect that can be observed in Figure 7 closely matches the way young children progressively construct new noun-verb associations.

One difference, however, is that the effect we observe is not centered around verbs. Although we presented generalization data obtained with ACTIONS `ate` and `loved` (within-verb and between-verbs generalization), the effect we described could have been observed with AGENTs, and OBJECTs too, because our model is symmetric. In contrast, the effect described by Tomasello is specific to verbs. However, one argument could favor linear RAAM as a model of human generalization. There exists an over-representation of nouns and an under-representation of verbs in children's early vocabulary, compared to their frequency in input speech (Gentner, 1978). Several hypotheses have been proposed in order to explain this asymmetry. One hypothesis is that since verbs describe actions, they are cognitively more complex than nouns (Gentner, 1978). In addition, verbs often describe transient actions, while nouns describe discrete objects in ostensive contexts, which might facilitate word-referent assocation (Tomasello and Kruger, 1992). Another hypothesis is that some preliminary noun knowledge is necessary in order to be able to acquire verb syntax (Gillette et al., 1999); in contrast, nouns can be acquired and readily used without syntax, which should be achievable earlier. Although these hypotheses are still debated, they all place the origin of the noun-verbs asymmetry outside the device that is used for their representation, and are therefore compatible with a representation system like linear RAAM, where actions and objects have the same type of representation.

6.6 Interpretation

Word-order preservation, as well as the early preference for context-specific generalization in our network, can be explained by linear combinations of reconstruction errors. We will demonstrate this on a simple example. Consider the following training set :

`( loved John Pat )`	(P1)
`( loved John Mary )`	(P2)
`( loved man Mary )`	(P3)
`( saw man Mary )`	(P4)

The two following propositions are linear combinations of elements of the training set :

`( loved man Pat )`	(P1)-(P2)+(P3)
`( saw man Pat )`	(P1)-(P2)+(P4)

These new formed propositions are consistent with word order. However, `( loved man Pat )` is within-verb generalization, while `( saw man Pat )` is between-verbs generalization, because `Pat` never co-occurs with `saw` in the training set. We will show on this example that within-verb generalization is easier to perform than between-verbs generalization. The key point of our demonstration is that the reconstruction error associated with a triple is higher when the constituents of this triple do not co-occur in the training examples.

We first observe that linear combinations preserve word order. This is because linear combinations of trees do not change the order of their constituents. Reciprocally, word-order violations cannot result from linear combinations of word-order consistent propositions. For example, the word-order violating propositions `( Pat man Mary)`, `( saw Pat Mary )`^{Note7} will not be formed by linear combinations of the above training examples.

Now consider that a ternary linear RAAM is trained on the above set (and only it). We consider the reconstruction error performed when decoding the above triples. The error made in decoding each proposition is a vector, that has three fields, LEFT, MIDDLE and RIGHT, for ACTION, AGENT and OBJECT respectively. Let $\mathbf{e}_L$, $\mathbf{e}_M$, $\mathbf{e}_R$, denote the error vectors in LEFT, MIDDLE and RIGHT after decoding a proposition. Let $\mathbf{v}_{\texttt{loved}}$ denote the vector encoding of terminal `loved`, and so on for other terminals. Let $\mathbf{L}$, $\mathbf{M}$ and $\mathbf{R}$ denote the square matrices of synaptic weights from LEFT, MIDDLE and RIGHT to WHOLE, respectively.

The reconstruction error of a linear combination of structures is equal to the corresponding linear combination of reconstruction errors of its terms. Hence, the error vectors that are in the LEFT, MIDDLE and RIGHT fields of the reconstruction error of proposition `( loved man Pat )` are :

$$\begin{cases} \mathbf{e}_{L\ \texttt{(loved man Pat)}} = \mathbf{e}_{L\ \texttt{(loved John Pat)}} - \mathbf{e}_{L\ \texttt{(loved John Mary)}} + \mathbf{e}_{L\ \texttt{(loved man Mary)}} \\ \mathbf{e}_{M\ \texttt{(loved man Pat)}} = \mathbf{e}_{M\ \texttt{(loved John Pat)}} - \mathbf{e}_{M\ \texttt{(loved John Mary)}} + \mathbf{e}_{M\ \texttt{(loved man Mary)}} \\ \mathbf{e}_{R\ \texttt{(loved man Pat)}} = \mathbf{e}_{R\ \texttt{(loved John Pat)}} - \mathbf{e}_{R\ \texttt{(loved John Mary)}} + \mathbf{e}_{R\ \texttt{(loved man Mary)}} \end{cases}$$

Similarly, the vectors that are in the LEFT, MIDDLE and RIGHT fields of the reconstruction error for proposition `( saw man Pat )` are :

$$\begin{cases} \mathbf{e}_{L\ \texttt{(saw man Pat)}} = \mathbf{e}_{L\texttt{(loved John Pat)}} - \mathbf{e}_{L\texttt{(loved John Mary)}} + \mathbf{e}_{L\texttt{(saw man Mary)}} \\ \mathbf{e}_{M\ \texttt{(saw man Pat)}} = \mathbf{e}_{M\texttt{(loved John Pat)}} - \mathbf{e}_{M\texttt{(loved John Mary)}} + \mathbf{e}_{M\texttt{(saw man Mary)}} \\ \mathbf{e}_{R\ \texttt{(saw man Pat)}} = \mathbf{e}_{R\texttt{(loved John Pat)}} - \mathbf{e}_{R\texttt{(loved John Mary)}} + \mathbf{e}_{R\texttt{(saw man Mary)}} \end{cases}$$

The learning rule attempts to minimize the mean squared error associated to the structures of the training set. Hence, the norms of the corresponding error vectors $[\mathbf{e}_L; \mathbf{e}_M; \mathbf{e}_R]$ will be jointly minimized. If these norms become sufficiently small, then the reconstruction errors of `( loved man Pat )` and `( saw man Pat )` will become small too, and these linear combinations will be successfully encoded and decoded.

In general, the reconstruction error of a linear combination of elements of the training set will be higher than the reconstruction error of its constituents. However, reconstruction errors might cancel out in certain cases. In order to see this, we develop the preceding expressions using the synaptic weight matrices **L**, **M** and **R**. For the LEFT field of each reconstruction, this yields:

$$\begin{cases} \mathbf{e}_{L\ \texttt{(loved John Pat)}} = \mathbf{v}_{\texttt{loved}} - \mathbf{L}'\mathbf{L}\mathbf{v}_{\texttt{loved}} - \mathbf{L}'\mathbf{M}\mathbf{v}_{\texttt{John}} - \mathbf{L}'\mathbf{R}\mathbf{v}_{\texttt{Pat}} \\ \mathbf{e}_{L\ \texttt{(loved John Mary)}} = \mathbf{v}_{\texttt{loved}} - \mathbf{L}'\mathbf{L}\mathbf{v}_{\texttt{loved}} - \mathbf{L}'\mathbf{M}\mathbf{v}_{\texttt{John}} - \mathbf{L}'\mathbf{R}\mathbf{v}_{\texttt{Mary}} \\ \mathbf{e}_{L\ \texttt{(loved man Mary)}} = \mathbf{v}_{\texttt{loved}} - \mathbf{L}'\mathbf{L}\mathbf{v}_{\texttt{loved}} - \mathbf{L}'\mathbf{M}\mathbf{v}_{\texttt{man}} - \mathbf{L}'\mathbf{R}\mathbf{v}_{\texttt{Mary}} \\ \mathbf{e}_{L\ \texttt{(saw man Mary)}} = \mathbf{v}_{\texttt{saw}} - \mathbf{L}'\mathbf{L}\mathbf{v}_{\texttt{saw}} - \mathbf{L}'\mathbf{M}\mathbf{v}_{\texttt{man}} - \mathbf{L}'\mathbf{R}\mathbf{v}_{\texttt{Mary}} \end{cases}$$

For the MIDDLE field, this yields :

$$\begin{cases} \mathbf{e}_{M\ \texttt{(loved John Pat)}} = \mathbf{v}_{\texttt{John}} - \mathbf{M}'\mathbf{L}\mathbf{v}_{\texttt{loved}} - \mathbf{M}'\mathbf{M}\mathbf{v}_{\texttt{John}} - \mathbf{M}'\mathbf{R}\mathbf{v}_{\texttt{Pat}} \\ \mathbf{e}_{M\ \texttt{(loved John Mary)}} = \mathbf{v}_{\texttt{John}} - \mathbf{M}'\mathbf{L}\mathbf{v}_{\texttt{loved}} - \mathbf{M}'\mathbf{M}\mathbf{v}_{\texttt{John}} - \mathbf{M}'\mathbf{R}\mathbf{v}_{\texttt{Mary}} \\ \mathbf{e}_{M\ \texttt{(loved man Mary)}} = \mathbf{v}_{\texttt{man}} - \mathbf{M}'\mathbf{L}\mathbf{v}_{\texttt{loved}} - \mathbf{M}'\mathbf{M}\mathbf{v}_{\texttt{man}} - \mathbf{M}'\mathbf{R}\mathbf{v}_{\texttt{Mary}} \\ \mathbf{e}_{M\ \texttt{(saw man Mary)}} = \mathbf{v}_{\texttt{man}} - \mathbf{M}'\mathbf{L}\mathbf{v}_{\texttt{saw}} - \mathbf{M}'\mathbf{M}\mathbf{v}_{\texttt{man}} - \mathbf{M}'\mathbf{R}\mathbf{v}_{\texttt{Mary}} \end{cases}$$

For the RIGHT field, this yields :

$$\begin{cases} \mathbf{e}_{R\ \texttt{(loved John Pat)}} = \mathbf{v}_{\texttt{Pat}} - \mathbf{R}'\mathbf{L}\mathbf{v}_{\texttt{loved}} - \mathbf{R}'\mathbf{M}\mathbf{v}_{\texttt{John}} - \mathbf{R}'\mathbf{R}\mathbf{v}_{\texttt{Pat}} \\ \mathbf{e}_{R\ \texttt{(loved John Mary)}} = \mathbf{v}_{\texttt{Mary}} - \mathbf{R}'\mathbf{L}\mathbf{v}_{\texttt{loved}} - \mathbf{R}'\mathbf{M}\mathbf{v}_{\texttt{John}} - \mathbf{R}'\mathbf{R}\mathbf{v}_{\texttt{Mary}} \\ \mathbf{e}_{R\ \texttt{(loved man Mary)}} = \mathbf{v}_{\texttt{Mary}} - \mathbf{R}'\mathbf{L}\mathbf{v}_{\texttt{loved}} - \mathbf{R}'\mathbf{M}\mathbf{v}_{\texttt{man}} - \mathbf{R}'\mathbf{R}\mathbf{v}_{\texttt{Mary}} \\ \mathbf{e}_{R\ \texttt{(saw man Mary)}} = \mathbf{v}_{\texttt{Mary}} - \mathbf{R}'\mathbf{L}\mathbf{v}_{\texttt{saw}} - \mathbf{R}'\mathbf{M}\mathbf{v}_{\texttt{man}} - \mathbf{R}'\mathbf{R}\mathbf{v}_{\texttt{Mary}} \end{cases}$$

Each of these reconstruction errors is a sum of four terms. The first term is the terminal vector itself. The three other terms are different contributions to its reconstruction, that come from the three fields. One of these contributions comes from the same field as the terminal itself (the one where the same matrix and its transpose is used). We call this term self-reconstruction. For example, the self reconstruction of $\mathbf{v}_{\texttt{loved}}$ is $\mathbf{L}'\mathbf{L}\mathbf{v}_{\texttt{loved}}$ in each of the above cases. The two additional terms are contributions from other fields. We call them contextual reconstructions.

We may express the above reconstruction errors as the sum of the reconstruction error from one element of the training set, plus some contextual contributions. For the proposition `( saw man Pat )`, this yields :

$$\left\{\begin{array}{lcllll} \mathbf{e}_{L\ \texttt{(loved man Pat)}} & = & \mathbf{e}_{L\ \texttt{(loved man Mary)}} & + & \mathbf{L}'\mathbf{R} & (\mathbf{v}_{\texttt{Mary}} - \mathbf{v}_{\texttt{Pat}}) \\ \mathbf{e}_{M\ \texttt{(loved man Pat)}} & = & \mathbf{e}_{M\ \texttt{(loved man Mary)}} & + & \mathbf{M}'\mathbf{R} & (\mathbf{v}_{\texttt{Mary}} - \mathbf{v}_{\texttt{Pat}}) \\ \mathbf{e}_{R\ \texttt{(loved man Pat)}} & = & \mathbf{e}_{R\ \texttt{(loved John Pat)}} & + & \mathbf{R}'\mathbf{M} & (\mathbf{v}_{\texttt{John}} - \mathbf{v}_{\texttt{man}}) \end{array}\right.$$

Similarly, for the proposition `( saw man Pat )`, we obtain the following decomposition :

$$\left\{\begin{array}{lcllllll} \mathbf{e}_{L\ \texttt{(saw man Pat)}} & = & \mathbf{e}_{L\texttt{(saw man Mary)}} & + & \mathbf{L}'\mathbf{R} & (\mathbf{v}_{\texttt{Pat}} - \mathbf{v}_{\texttt{Mary}}) & & \\ \mathbf{e}_{M\ \texttt{(saw man Pat)}} & = & \mathbf{e}_{M\texttt{(saw man Mary)}} & + & \mathbf{M}'\mathbf{R} & (\mathbf{v}_{\texttt{Pat}} - \mathbf{v}_{\texttt{Mary}}) & & \\ \mathbf{e}_{R\ \texttt{(saw man Pat)}} & = & \mathbf{e}_{R\texttt{(loved John Pat)}} & + & \mathbf{R}'\mathbf{M} & (\mathbf{v}_{\texttt{John}} - \mathbf{v}_{\texttt{man}}) & + & \mathbf{R}'\mathbf{L} \quad (\mathbf{v}_{\texttt{loved}} - \mathbf{v}_{\texttt{saw}}) \end{array}\right.$$

The decomposition of $\mathbf{e}_{R\texttt{(saw man Pat)}}$ involves two contextual contributions. It is not possible to find a better simplification with only one contextual contribution. This is because in the training examples, `Pat` never occur together with `saw` or `man`. More generally, the number of contextual contributions is the number of fields that have to be replaced in order to obtain an element of the training set. Hence, within-verb generalization will generate reconstruction errors with only one contextual contribution, while between-verbs generalization will generate reconstruction errors with two contextual contributions.

The number of contextual contributions reflects the number of simplifications that occur when reconstruction errors are linearly combined. A higher number of contextual contributions means that less simplifications occur, and in this case the overall reconstruction error will be higher. This explains why the linear RAAM needs more training time in order to use learned words in new contexts.

7 Conclusion

We presented a powerful modification of RAAM, that uses linear units and is trained with Oja's learning rule. This linear RAAM represents novel structures by linearly combining learned examples. This combinatorial capability results in a very high generalization capacity (up to 100 times that of the original RAAM in our experiments). Interestingly, generalization preserves certain properties of the training set; the network is sensitive to word order and to context. This relates our model to recent theories of language acquisition (Tomasello, 1992).

Our analysis of how the model generalizes makes it possible to partially predict which new structures can be represented. For example, considering Fodor's challenge, it is unlikely that a linear RAAM trained to represent the phrase "John loves the girl" will immediately be able to represent "the girl loves John". This is because, in the absence of real world referents for "John" and "the girl", the network is unable to understand the common properties between these objects. However, if the same network is further exposed to other phrases, where "the girl" and "John" are used indifferently as subjects and as objects of transitive verbs, then it will learn the common properties of "the girl" and "John", and it will be able to represent "the girl loves John". In this sense, the representations in a linear RAAM are systematic. In this account of systematicity, the model learns the properties of objects based on their relations in the training set.

In the classical RAAM and its variants, systematicity is based on the properties of the neural code; similar concepts are represented by topologically close vectors, which enables relevant generalization (Pollack, 1990; Niklasson and van Gelder, 1994). Bodén and Niklasson (2000) recently proposed an extension of RAAM that learns the properties of objects, based on their relations within the training set. In their model, systematicity is achieved by learning the encodinf of terminals. Although our model also learns relations between objects, we deliberately precluded the effects of terminal encoding, because our aim was to study how systematicity could result from linear combinations of structures. However, we also demonstrated that our model too is sensitive to the properties of the neural code. Hence, both strategies may be seen as dual, and it should be possible to use them in combination.

References

Baldi, P. and Hornik, K. (1988). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58.

Blank, D., Meeden, L., and Marshall, J. (1992). Exploring the symbolic/subsymbolic continuum: A case study of RAAM. In Dinsmore, J., editor, *The Symbolic and Connectionist Paradigms: Closing the Gap.*, pages 113–148. Lawrence Erlbaum Associates, New Jersey, USA.

Boden, M. and L., N. (2000). Semantic systematicity and context in connectionist networks. *Connection Science*, 12(2):1–31.

Brousse, O. (1993). *Generativity and Systematicity in Neural Network Combinatorial Learning.* PhD thesis, University of Colorado at Boulder.

Bryson, A. E. and Ho, Y.-C. (1975). *Applied Optimal Control.* John Wiley, N.Y.

Callan, R. E. and Palmer-Brown, D. (1997). (S)RAAM: An analytical technique for fast and reliable derivation of connectionist symbol structure representations. *Connection Science*, 9(2):139–160.

Chalmers, D. (1990). Syntactic transformations on distributed representations. *Connection Science*, 2:53–62.

Chrisman, L. (1991). Learning recursive distributed representations for holistic computation. *Connection Science*, 3(4):345–366.

Fodor, J. (1975). *The Language of Thought.* Crowell, New York.

Fodor, J. A. and Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: a critical analysis. *Cognition*, 28:3–71.

Gentner, D. (1978). On relational meaning : the acquisition of verb meaning. *Child Development*, 49:988–998.

Gillette, J., Gleitman, H., Gleitman, L., and Lederer, A. (1999). Human simulations of vocabulary learning. *Cognition*, 73:135–176.

Hadley, R. F. (1994). Systematicity in connectionist language learning. *Mind and Language*, 9(3):247–272.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46:47–75.

Hinton, G. E., McClelland, J. L., and Rumelhart, D. E. (1986). Distributed representations. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1: Foundations, pages 77–109. MIT Press, Cambridge, MA, USA.

Kwasny, S. and Kalman, B. (1995). Tail-recursive distributed representations and simple recurrent networks. *Connection Science*, 7:61–80.

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

Niklasson, L. and van Gelder, T. (1994). On being systematically connectionist. *Mind and Language*, 9:288–302.

Oja, E. (1989). Neural networks, principal components and subspaces. *International Journal of Neural Systems*, 1(1):61–68.

Plate, T. (1994). *Distributed Representations and Nested Compositional Structure*. PhD thesis, Department of Computer Science, University of Toronto.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, 46:77–105.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representation by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. Cambridge, MA: MIT Press.

Smolensky, P. (1987). The constituent structure of connectionist mental states: A reply to Fodor and Pylyshyn. *Southern Journal of Philosophy*, 26:137–159.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–216.

Soderstrom, T. and Stoica, P. (1989). *System Identification*. Prentice-Hall, N.J.

Sperduti, A. (1994). Labeling RAAM. *Connection Science*, 6(4):429–459.

Stolcke, A. and Wu, D. (1992). Tree matching with recursive distributed representations. Technical Report TR-92-025, International Computer Science Institute, Berkeley.

Tomasello, M. (1992). *First verbs : A case study in early grammatical development*. Cambridge University Press.

Tomasello, M. (2000). Do young children have adult syntactic competence? *Cognition*, 74:209–253.

Tomasello, M. and Kruger, A. (1992). Joint attention on actions: acquiring verbs in ostensive and non-ostensive contexts. *Journal of Child Language*, 19:311–334.

Touretzky, D. (1990). BOLTZCONS: Dynamic symbol structures in a connectionist network. *Artificial Intelligence*, 46:5–46.

Voegtlin, T. (2000). Learning principal components in a contextual space. In Verleysen, M., editor, *Proceedings of ESANN'2000*, pages 359–364. D Facto, Bruxelles.

Voegtlin, T. and Dominey, P. (2001). Learning high-degree sequences in a linear network. In *Proceedings of the IJCNN'2001*, volume 1, pages 940–944.

Notes

Note 1: Training a RAAM with backpropagation is a moving-target problem as well.

Note 2: In general, it is assumed that the input vector used in PCA has zero mean, which can be achieved by subtraction of a constant vector. For the results we present here, however, this hypothesis is not crucial. Hence, in order to exactly reproduce the experimental protocol in Pollack (1990), we did not center the input vector.

Note 3: Starting from a higher value for η and decreasing it during learning might reduce the number of necessary iterations.

Note 4: Callan and Palmer-Brown (1997) encoded this dataset with only 11 neurons. This difference is due to the fact that they centered the data around zero before proceeding with the PCA.

Note 5: Meaningful generalization is a type of consistency. In testing consistency, it is necessary to demonstrate that inconsistent propositions cannot be represented. Hence, although Pollack's observation that `(loved x y)` can be represented for any `x` and `y` in the set { `John Mary Pat man` } is relevant for systematicity, it is not sufficient to demonstrate consistency. A consistency test would require that meaningless propositions cannot be represented.

Note 6: Of course we could have designed a training set that does not have this drawback. However, we preferred to use an existing example.

Note 7: This proposition violates word order given the restricted training set.

Figures

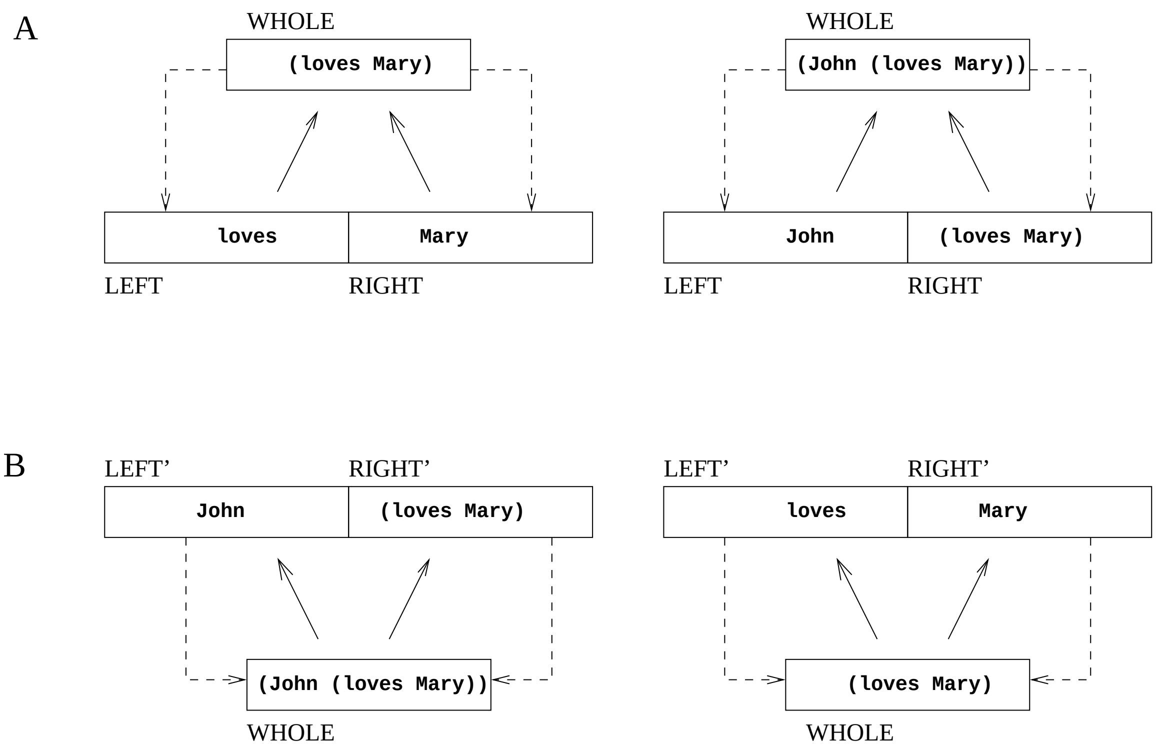
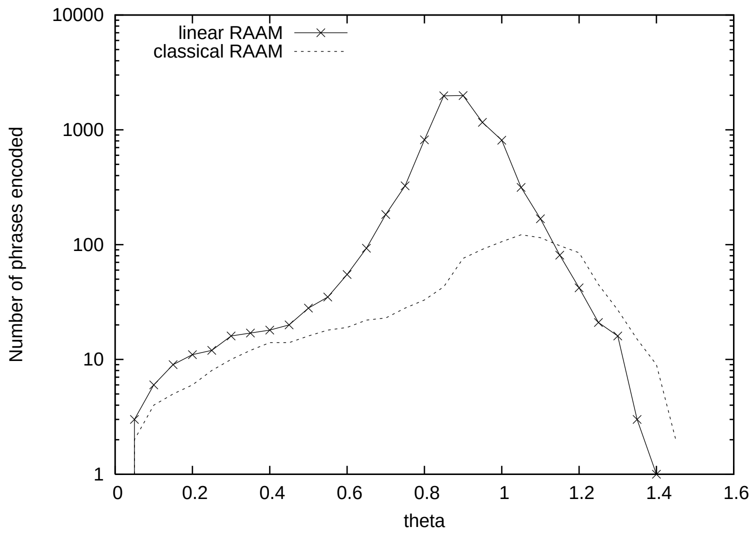
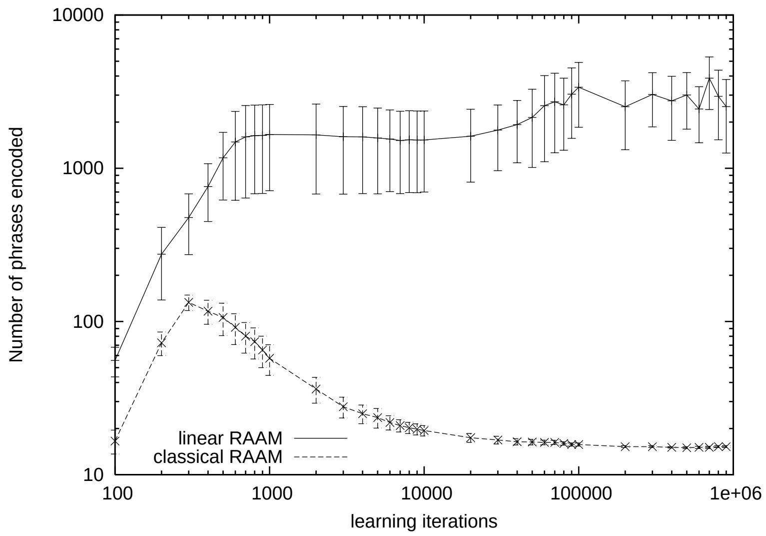


Figure 1:



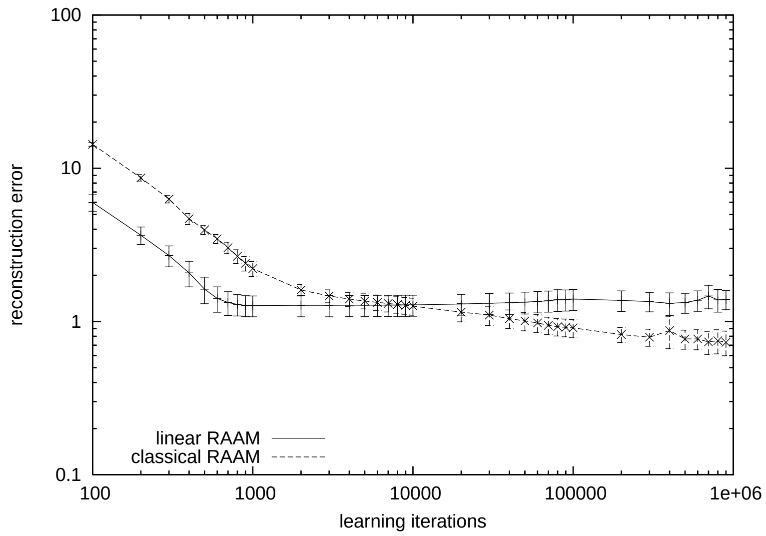
linear RAAM
classical RAAM
Number of phrases encoded
theta
10000
1000
100
10
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6

Figure 2:



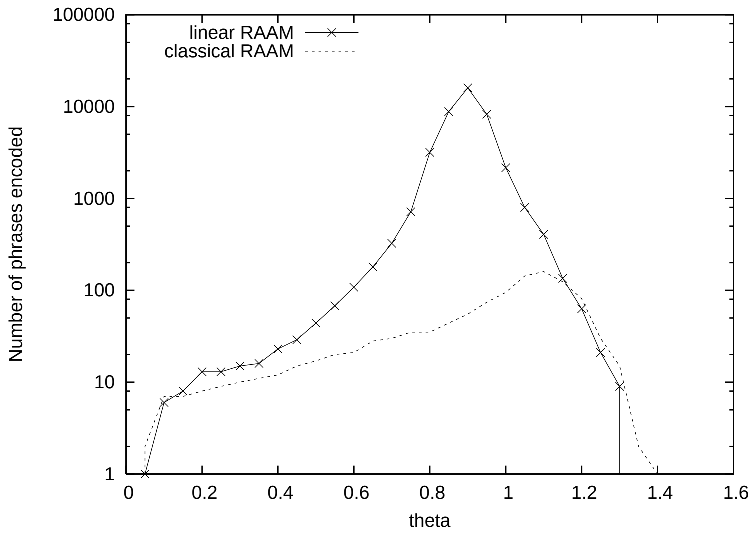
10000
1000
100
10
Number of phrases encoded
linear RAAM
classical RAAM
100
1000
10000
100000
1e+06
learning iterations

Figure 3:



100
10
1
0.1
reconstruction error
100
1000
10000
100000
1e+06
learning iterations
linear RAAM
classical RAAM

Figure 4:



linear RAAM
classical RAAM
Number of phrases encoded
theta
100000
10000
1000
100
10
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6

Figure 5:

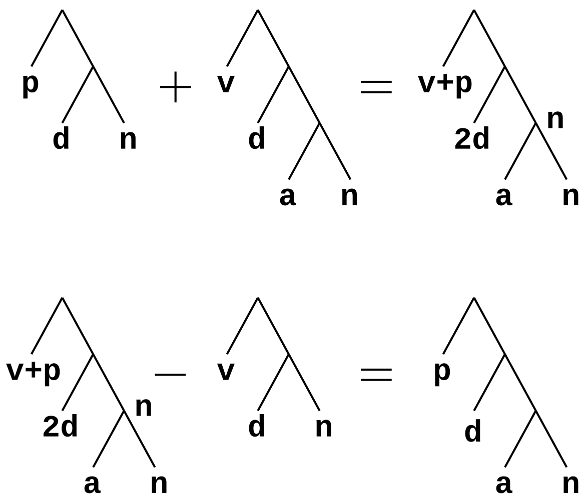
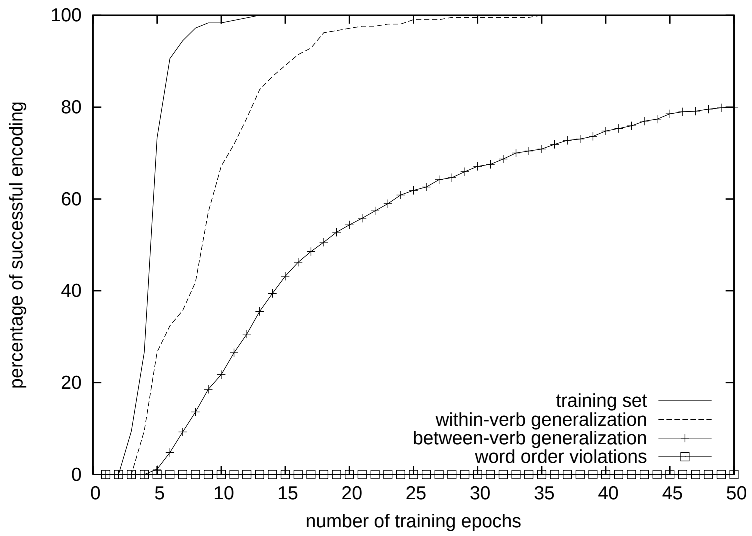


Figure 6:



percentage of successful encoding
number of training epochs
training set
within-verb generalization
between-verb generalization
word order violations
0
20
40
60
80
100
0
5
10
15
20
25
30
35
40
45
50

Figure 7:

Figure Legends

Figure 1: Function of a binary RAAM. Dashed arrows indicate recurrent one-to-one connections, and continuous arrows indicate full connectivity connection matrices. A: The encoder generates a compact representation of the complex object `(John (loves Mary))` in two steps. First, a representation of `(loves Mary)` is formed (left). Pattern of activities in LEFT and RIGHT represent `loves` and `Mary`, respectively, and the resulting pattern in WHOLE is a representation of `(loves Mary)`. Then, the representation of `(loves Mary)` is recycled in LEFT through recurrent connections, and a new pattern is put in RIGHT, that represents `John` (right). This generates a representation of `(John (loves Mary))` in WHOLE. B: The decoder generates a reconstruction of the atomic constituents of the complex structure `(John (loves Mary))`. First, reconstructions of `John` and of `(loves Mary)` are generated (left). Second, the representation of the non terminal `(loves Mary)` is copied in WHOLE with recurrent connections, and its constituents, `loves` and `Mary`, are decoded in vectors LEFT' and RIGHT' (right).

Figure 2: Number of phrases successfully encoded by a 20-10-20 RAAM, trained using back-propagation (dashed line) or using linear units and Oja's rule (continuous line). Networks were trained to represent Pollack's corpus (7 phrases). Generalization performance was measured for different values of the threshold θ between 0 and 1.6. Performance is reported using a logarithmic scale.

Figure 3: Effect of training on performance. The number of successfully encoded phrases averaged over 11 experiments is plotted. Different random initial weight were used in each experiment. Error bars indicate the standard deviation. A logarithmic scale is used for both performance and time.

Figure 4: Effect of training on reconstruction error. Error is averaged over all the vectors that are built during one presentation of the training set. Results are averaged over 11 experiments with different random initial weights. Error bars indicate the standard deviation.

Figure 5: Number of phrases successfully encoded by a RAAM of size 20-10-20, trained using back-propagation (dashed line) or using linear units and Oja's rule (continuous line). Networks were trained to represent the 14 phrases of Table 4. After training, generalization performance was measured for different values of the threshold θ between 0 and 1.6. Performance is reported using a logarithmic scale.

Figure 6: Addition and subtraction of complex structures using superpositions of binary trees. Top: The addition of `(p(dn))` and `(v(d(an)))` does not encode a phrase; the corresponding tree is not valid. Bottom: The result of `(v(d(an)))+(p(dn))-(v(dn))` encodes the phrase `(p(d(an)))`

Figure 7: Generalization performance on simple propositions built with the verbs `loved` and `ate`. The distinction is made between four types of propositions : Propositions found in the training set, new formed propositions where a verb is used with the same nouns as it is used with in the training set ("within-verb generalization"), new formed propositions where a verb is used with nouns that have been used with other verbs in the training set ("between-verb generalization"), and propositions that have word-order violations.

Table 1: Extended corpus used by Pollack (1990).

Training examples.	Additional structures contained in the training examples.
(d(a(a(an))))	(dn)
((dn)(p(dn)))	(d(an))
(v(dn))	(v(p(dn)))
(p(d(an)))	(v(d(an)))
((dn)v)	(p(dn))
((dn)(v(d(an))))	(an)
((d(an))(v(p(dn))))	(a(an))
	(a(a(an)))

Table 2: Simple context-free grammar used to generate well-formed syntactic trees. Source: Pollack (1990).

Sentence	*Noun Phrase*	*Verb Phrase*	*Prep. Phrase*	*Adj. Phrase*
S → NP VP	NP → d AP	VP → v NP	PP → p NP	AP → a AP
S → NP v	NP → d n	VP → v PP		AP → a n
	NP → NP PP			

Table 3: New formed structures that can be represented by a Classical 20-10-20 RAAM. Source: Pollack (1990).

```
(da)
(va)
(vn)
(vv)
(((dn)(p(dn)))n)
(((dn)(p(dn)))(d(an)))
((dn)(((dn)(p(dn)))(d(an))))
(((dn)(p(dn)))((dn)(p(dn))))
(((dn)(p(dn)))((d(an))(p(dn))))
((dn)(((dn)(p(dn)))((d(an))(p(dn)))))
(((dn)(p(dn)))(((dn)(p(dn)))(d(an))))
(((dn)(p(dn)))((dn)(p(dn)))((d(an))(p(dn))))
((d(an))(p(dn)))
((dn)(p(d(an))))
((d(an))(p(d(an))))
((v(dn))(p(dn)))
((v(d(an)))(p(dn)))
((v(dn))(p(d(an))))
((v(d(an)))(p(d(an))))
((dn)(v(dn)))
(((dn)(p(dn)))v)
((dn)(v(dn)(p(dn))))
(((dn)(p(dn)))(v(dn)))
((dn)(v((d(an))(p(dn)))))
((dn)(v((dn)(p(d(an))))))
(((dn)(p(dn)))(v(d(an))))
((dn)(v((d(an))(p(d(an))))))
(((dn)(p(dn)))(v((dn)(p(dn)))))
(((dn)(p(dn)))(v((dn)(p(d(an))))))
(((dn)(p(dn)))(v((d(an))(p(dn)))))
(((dn)(p(dn)))(v((d(an))(p(d(an))))))
```

Table 4: Training set used in the second experiment.

```
(a(a(a(a(an)))))
(d(a(a(a(an)))))
((d(a(a(an))))v)
((d(an))(p(dn)))
((d(an))(v(dn)))
((dn)(p(d(an))))
(((dn)(p(dn)))v)
((dn)(v(d(an))))
((dn)(v(p(dn))))
(p(d(a(a(an)))))
(p((dn)(p(dn))))
(v(d(a(a(an)))))
(v((dn)(p(dn))))
(v(p(d(a(an)))))
```

Table 5: Minimal value of the threshold θ that allows to successfully encode and decode the corpus, for different sizes of the representation. Representations with less than seven neurons could not correctly perform the task. For representations that use 12 or more neurons, the threshold can be chosen arbitrarily small.

number of neurons	7	8	9	10	11	12
minimal θ	0.53	0.44	0.44	0.34	0.16	≈ 0

Table 6: New formed structures by the linear RAAM, when the error threshold is zero.

(d(a(an)))
(a(a(a(an))))
((dn)(v(dn)))
(p(d(a(an))))
(v(d(a(an))))
(v(p(d(an))))
((dn)(p(d(an))))
(pv)
(vv)
((dn)(dn))
(p(p(dn)))
(p(v(dn)))
(v(v(dn)))
((dn)(d(an)))
(p(p(d(an))))
(p(v(d(an))))
(v(v(d(an))))
((dn)(d(a(an))))
((d(an))(p(p(dn))))

Table 7: Ternary representation of semantic propositions. Source : Pollack (1990)

```
( loved Pat Mary )
( loved John Pat )
( ( with saw telescope ) John ( on man hill ) )
( ( with ate chopsticks ) Mary spaghetti )
( ate Mary ( with spaghetti meat ) )
( ate Pat meat )
( knew Pat ( loved John Mary ) )
( thought Pat ( knew John ( loved Mary John ) ) )
( hoped Pat ( thought John ( ate Mary spaghetti ) ) )
( ( with hit ( mod telescope long ) ) John man )
( hit John ( with man ( mod telescope long ) ) )
( hoped Pat ( saw ( with man telescope ) Pat ) )
( hit Pat ( is man ( thought man ( loved Mary John ) ) ) )
( saw ( is ( mod man short ) ( thought man ( saw man John ) ) ) Pat )
```

Table 8: 16-bits encoding of words used by Pollack (1990) in the propositional experiment.

word	THING	HUMAN	PREP	ADJ	VERB
hill	1 0 0 0				
street	1 0 0 1				
telescope	1 0 1 0				
chopsticks	1 0 1 1				
meat	1 1 0 0				
spaghetti	1 1 0 1				
man		1 0 0			
John		1 0 1			
Mary		1 1 0			
Pat		1 1 1			
mod			1 0 0		
with			1 0 1		
on			1 1 0		
long				1 0	
short				1 1	
is					1 0 0 0
knew					1 0 0 1
hoped					1 0 1 0
thought					1 0 1 1
loved					1 1 0 0
hit					1 1 0 1
ate					1 1 1 0
saw					1 1 1 1

Table 9: Triples based on the verbs `loved` and `ate`.

Training set	Between-verb generalization	Word-order violations
`( ate Pat meat )`	`( ate Pat Pat )`	`( ate meat Mary )`
`( ate Mary spaghetti )`	`( ate Pat John )`	`( ate meat John )`
`( loved John Pat )`	`( ate Pat Mary )`	`( ate meat Pat )`
`( loved John Mary )`	`( ate Mary Mary )`	`( ate meat meat )`
`( loved Pat Mary )`	`( ate Mary John )`	`( ate meat spaghetti )`
`( loved Mary John )`	`( ate Mary Pat )`	`( loved meat Mary )`
	`( ate John John )`	`( loved meat John )`
	`( ate John Mary )`	`( loved meat Pat )`
	`( ate John Pat )`	`( loved meat meat )`
	`( ate John meat )`	`( loved meat spaghetti )`
	`( ate John spaghetti )`	
	`( loved Pat meat )`	
Within-verb generalization	`( loved Mary meat )`	
`( ate Mary meat )`	`( loved John meat )`	
`( ate Pat spaghetti )`	`( loved Pat spaghetti )`	
`( loved John John )`	`( loved Mary spaghetti )`	
`( loved Pat Pat )`	`( loved John spaghetti )`	
`( loved Pat John )`	`( ate spaghetti Mary )`	
`( loved Mary Mary )`	`( ate spaghetti John )`	
`( loved Mary Pat )`	`( ate spaghetti Pat )`	
	`( ate spaghetti spaghetti )`	
	`( ate spaghetti meat )`	
	`( loved spaghetti Mary )`	
	`( loved spaghetti John )`	
	`( loved spaghetti spaghetti )`	
	`( loved spaghetti meat )`	