INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center Street • Suite 600 • Berkeley, California 94704 • 1-415-642-4274 • FAX 1-415-643-7684

Learning Feature-based Semantics with Simple Recurrent Networks

Andreas Stolcke[1]

TR-90-015

April 1990

Abstract

The paper investigates the possibilities for using simple recurrent networks as transducers which map sequential natural language input into non-sequential feature-based semantics. The networks perform well on sentences containing a single main predicate (encoded by transitive verbs or prepositions) applied to multiple-feature objects (encoded as noun-phrases with adjectival modifiers), and shows robustness against ungrammatical inputs. A second set of experiments deals with sentences containing embedded structures. Here the network is able to process multiple levels of sentence-final embeddings but only one level of center-embedding. This turns out to be a consequence of the network's inability to retain information that is not reflected in the outputs over intermediate phases of processing. Two extensions to Elman's [9] original recurrent network architecture are introduced.

[1]The author is supported by an IBM Graduate Fellowship.

1 Introduction

Since their introduction by Elman [9] backpropagation networks with one time step of hidden layer recurrence (simple recurrent networks, SRNs) have been extensively exploited as a simple, efficient, and yet surprisingly powerful architecture for dealing with sequential input and output patterns. The sequential nature of these networks makes them a natural candidate for natural language processing tasks, as has been variously demonstrated [1, 22, 23, 11, 13, 19].

The application of SRNs presented here purports to use a task and training environment that is 'natural' in several ways. First, the goal for the network is to act as a surface-to-semantic transducer, i.e. to map a sequence of words into a representation of the corresponding semantics. This contrasts with other approaches which have the network predict next words [11], or fill in the unaltered word patterns into slots [19]. Second, the representations are sequential at the input (word) level, but parallel at the output (semantic) level. This is intended to be a rough approximation of a real-world learning situation in which an observer tries to infer word and sentence meanings by matching low-bandwidth linguistic input against high-bandwidth input from other modalities (such as vision). This idealizes a situation where there are no temporal clues concerning the correlation between input elements and target elements. This approach contrasts with others where the network is trained to map the sequential input into sequences of outputs with a temporal structure that is essentially isomorphic [22]. It also differs from schemes which present inputs sequentially, but encode some of the sequential structure redundantly in the input patterns themselves, or preprocess inputs into phrase-level chunks, thus avoiding even simple syntactic complexities such as adjective-noun combinations and function words [23].

The learning environment is largely consistent with the miniature language acquisition problem recently proposed as a framework for interdisciplinary research by Feldman et. al [12], although it addresses only a subset of the learning task proposed by the authors.

S	$\rightarrow$	NP |NP VP
VP	$\rightarrow$	VI PP
	$\rightarrow$	VT NP
PP	$\rightarrow$	P1 NP
P1	$\rightarrow$	{Mod} P
NP	$\rightarrow$	Det N1
N1	$\rightarrow$	Adj* N
VI	$\rightarrow$	*is*
VT	$\rightarrow$	*touches*
Det	$\rightarrow$	*a*
N	$\rightarrow$	*circle* |*square* |*triangle*
Adj	$\rightarrow$	*small* |*medium* |*large* | *light* |*dark*
P	$\rightarrow$	*above* |*below* | *to the left of* |*to the right of*
Mod	$\rightarrow$	*far*

Figure 1: Grammar generating language fragment.

2 The learning task

The semantic domain selected for our experiments consists of two-dimensional geometric objects (circles, squares, triangles), unary predicates of these objects (size, shade), and spatial relations between objects (left, right, above, below, contact). The subset of English sentences used to describe this domain is generated by the phrase structure grammar in Figure 1.[1]

The mapping from entities in the semantic domain to verbalizations is straightforward. Objects are described by noun phrases (NPs) with up to two adjectives, e.g.

a small circle

Spatial relations are expressed as complete sentences, with the predicate expressed either by a transitive verb (*touches*) or the head of a prepositional phrase (PP), optionally modified by *far*:

[1] Note that the fragment represents a a slightly restricted version of the L_0 language specified by Feldman et. al [12].

a light circle touches a small square
a large dark square is far to the left of
a small triangle

Note that isolated NPs are allowed as sentences, denoting minimal subsets of the domain (i.e., objects and unary predicates).

On the input side, words are encoded by 19 orthogonal vectors (this includes non-function words such as *a, to, the, of*). This representation, although wasteful, was chosen because it does not encode any similarities between word functions given a priori. Such similarities will have to be detected and re-encoded by the network using its hidden unit activations.

Note that the language fragment as considered so far is not even properly context-free (in fact, it is finite), and each sentence conveys at most a binary predicate and two arguments, plus possibly some unary predications (an extension to a real context-free fragment with multiple predicates per sentence is discussed below). Therefore sentence semantics can be encoded in a 22-bit, fixed-width feature vector of the following form

$$\overbrace{\underbrace{\mathtt{T\ L\ R\ A\ B}}_{\text{relation}}\ \underbrace{\mathtt{F}}_{\text{mod}}}^{\text{Predicate}}\ \overbrace{\underbrace{\mathtt{C\ S\ T}}_{\text{shape}}\ \underbrace{\mathtt{S\ M\ L}}_{\text{size}}\ \underbrace{\mathtt{D\ L}}_{\text{shade}}}^{\text{Argument 1}}\ \overbrace{\underbrace{\mathtt{C\ S\ T}}_{\text{shape}}\ \underbrace{\mathtt{S\ M\ L}}_{\text{size}}\ \underbrace{\mathtt{D\ L}}_{\text{shade}}}^{\text{Argument 2}}$$

A bit takes on the value 1 if the corresponding semantic feature is present, and 0 otherwise. The semantics of a single object (described by an isolated NP) are encoded by activations in the 'Argument 1' slot with all other features turned off.

Initial experiments were conducted using Elman's original recurrent network architecture, depicted in Figure 2a. Word patterns were presented sequentially at the input layer, while the target semantics pattern was held constant for the duration of the entire sentence. The hidden layer was fed with the current word pattern as well as a copy of its own activations from the previous time step (initialized to zero at the beginning of each sentence). After each processing step backpropagation (with sum-of-squares error function) was applied to adjust all weights on-line.

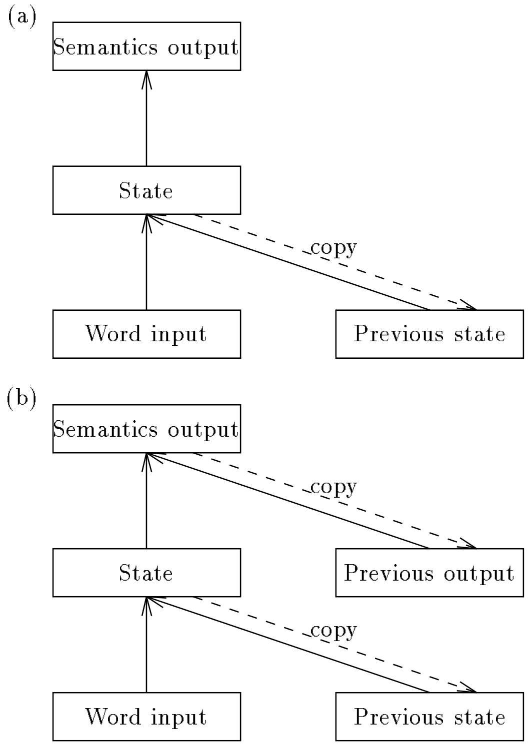

Figure 2: (a) Simple recurrent network architecture. Solid arrows represent normal weighted, learnable links (layers fully connected). Dashed arrows indicate transfer of activations from the previous time step. (b) Recurrent network enhanced to support incremental construction of output vectors.

As expected, the network trained in this way learned to build the output pattern incrementally, essentially turning on bits as soon as the corresponding words had been seen in the input and then keeping them on. To support this incremental behavior and unburden the hidden layer from the task of memorizing the outputs constructed so far, a small addition to the architecture was made, shown in Figure 2b. An additional recurrent loop at the output should enable the network to memorize and reuse previous outputs in a more straightforward manner.[2]

[2] But note that the connections from the 'output memory' back to the output layer still have to be learned as all others. The resulting recurrent architecture combines features of both the Elman and the Jordan [15] type.

Although the total number of links was not reduced this way, the smaller number of *hidden* units (15 instead of 25 in this experiment) sped up convergence.

As various researchers have noted [10, 7], backpropagation learning on large training sets with complex underlying structure is helped by presenting the training set incrementally, in stages of increasing size and/or complexity. Accordingly, we increased the complexity of the training set typically along the following lines: simple NPs (Det-N), one-adjective and two-adjective NPs, sentences containing simple NPs, sentences with complex NPs (note that all these are legal sentences in the grammar). At each stage, learning rates were gradually decreased from 0.1 to 0.0001, at which point training proceeded until the error leveled off; momentum was held constant at 0.3.

3 Simple Sentences: Results

The final training set consisted of 75% of all possible simple NP sentences (61 out of 81), 10% of all one-adjective NP sentences (203 out of 2025) and 3.3% of all two-adjective NP sentences (89 out of 2916). When training was stopped the average Hamming distance (number of wrong bits) between the target vectors and the output generated after having seen the last word in a sentence was 0.082 on the training set (chance performance is 5.7 for this set).[3] All errors were due to complex-NP sentences. The network generalized perfectly to the remaining 25% of simple NP sentences and produced an average Hamming distance of 1.07 (chance: 5.5) and 0.94 (chance: 7.5) on one-adjective and two-adjective NP sentences, respectively. (Performance figures are summarized in Table 1.)

A more qualitative evaluation of the generalizations the network has extracted from the training data can be obtained from studying performance on various kinds of ungrammatical input (not contained in training set).

[3]In computing Hamming distances, a output was allowed to be within 0.4 of the target value (0 or 1) to count as correct.

Figure 3a shows the incremental assembly of the feature vector while processing a grammatical input sentence. Figure 3b shows what happens when articles are omitted from NPs. Since articles don't carry any information in the current training data we would expect the net to deal well with this case from a semantic perspective. Still, any change in the sequential structure of input could potentially disrupt the succession of state transitions the network makes in order to arrive at its result. As can be seen, however, the 1-bits in the output are only marginally weaker than in the previous case.

A more striking example are sentences where the relation-encoding preposition or verb is omitted from the sentence (figure 3c). The network will still correctly associate the two NPs with the two argument slots in the output vector. This shows that the rules of English regarding relative ordering of subject and object have been learned. However, the missing verb does reduce the ability of the network to keep the argument semantics separated, as features encoding the object NP 'leak' into the subject slot. This interpretation is consistent with the case where the subject has been omitted from the sentence (Figure 3d). The object NP is mostly assigned to the subject slot (recall that the network is trained to assign single NP sentences to that slot), but the fact that the NP is preceded by the verb seems to be evidence for the network to assign the NP at least partly to the object slot.

A very subtle form of ungrammaticality arises from the conventional order of adjectives in noun phrases. Projected to our language fragment, size adjectives precede shade adjectives when both are present, and this order was consistently used in the training set (cf. Figure 3e). Figure 3f shows the effect of reversing that order. The misplaced second adjective causes some initial activation in the corresponding output unit, but is then 'forgotten' as further words are processed. This example shows that processing NPs is more complex than simply adding up all the features extracted from individual words (although this would have been a minimal solution to the learning task easy

Pattern set	Size	initial s.s.	initial Ham.	final s.s.	final Ham.	chance Ham.
training set (75%/10%/3.3%)	353	6.0	13.8	0.12	0.082	5.7
simple NP sentences (100%)	81	5.5	11.2	0.098	0	3.5
1-adj NP sentences (100%)	2025	7.0	15.9	0.65	1.07	5.5
2-adj NP sentences (50%)	1458	5.9	14.3	0.57	0.94	7.5

Table 1: Summary of network performance after training on simple sentences. The initial performance (with randomly initialized weights) is given for comparison, as well as the expected Hamming distance obtained by chance (guessing the most likely of the values 0 or 1). Both sum squared errors and Hamming distances are averages per sentence.

to implement with the given architecture). Instead, bit correlations and sequential structure interact in non-trivial ways to yield the output.[4]

Another interesting feature of the network's operation is that 'meaning' is non-local in time in the following sense. The network updates its outputs constantly according to the semantics expected from the current input. Contrary to a view held by 'lexical' theories of language and grammar (such as, e.g., Lexical Functional Grammar [16]) where meanings are localized in individual lexicon entries, the predicate 'left' is not locally attached to the word *left*. Instead, it is shared by the syntagmatic environment of *left*. As illustrated in Figure 3g, activation on the 'left' feature builds up as the function words *is*, *to*, *the*, and *of* are processed.

4 Embedded structures

Keeping the output targets static for the entire duration of a sentence is appealing as a training scheme because it poses an interesting learning problem (the extraction of relative order information from the input) and is relatively plausible. A severe disadvantage is that it can only deal with semantic feature vectors of a fixed length. Even a single sentence, however, can encode arbitrarily many propositions or 'frames' if we make the following small addition to the grammar in Figure 1:

$$\text{NP} \quad \rightarrow \quad \text{Det N PP}$$

This allows for arbitrarily long chains of reduced relative clauses

a circle below a square
a triangle touches a circle below a circle to the left of a square ...[5]

Clearly, to represent the semantics of such sentences, the approach of spreading out the complete sentence semantics in space will not do. Ideally we would like to adopt a representation that can deal with hierarchical structures in a completely general way, which is yet to be discovered for connectionist models. We have therefore adopted a compromise solution which tries to preserve the spirit of the original scheme while still allowing us to get an idea of network performance on embedded structures.

We use the time dimension to multiplex the sentence semantics at the frame level, i.e. as an embedded clause is processed its semantics replace the previous feature vector in the output layer. Thus, when processing

(1) *a triangle touches a circle below a square*

[4] Although our model does not claim to model linguistic performance at this level of detail, one could object that the network is too restrictive compared to humans. Non-conventional adjective order would hardly prevent a speaker from making sense of a noun phrase. On the other hand, adjective order is not as strictly enforced in real language use as it was in our training set.

[5] For our purposes we ignore the alternative reading of such a sentence where the iterated embedded clauses are all attached to the first NP.

the network is trained to produce the semantics of 'triangle touches circle' up to the word *circle*. Following that the target is changed to 'circle below square'. Similarly, for

(2) *a triangle to the left of a circle touches a square*

the target is 'triangle touches square' for the entire sentence except for the duration of *to the left of a circle*, when it becomes 'triangle left of circle'.[6]

To accomplish this task successfully the network has to deal with at least two potential difficulties. First, the material in embedded clauses has to be kept separate from the surrounding context. In particular, in sentence (2) the network must not mistake the NP immediately preceding the verb for the subject of the main clause. Secondly, sentences such as (1) require that a semantic pattern switch position in the output vector, since 'circle' first appears as object, but then becomes the subject of the embedded clause. This creates a type of discontinuity in the output vector the network did not have to cope with in the previous experiment.

After initial experiments with the kind of networks shown in Figure 2 a further modification was introduced, as depicted in Figure 4. Its structure anticipates the need for a finite state control, directing the information passing through the network. The left half of the network operates and is trained as a next-word predictor, just like originally conceived by Elman [9]. This training setup is known to build hidden layer representations which are closely related to the finite-state structure of the input domain [4]. The design is such that the representations in this 'state' feedback loop can then control the information flowing through a separate 'memory' feedback loop and the semantics output, but not vice-versa. Note that the representations in the 'state' hidden layer are influenced by the constraints of the semantic part of the output during backpropagation, but its performance after learning is design to be independent of the semantics component.

An informal comparison seemed to favor this architecture on the basis of faster convergence and lower final error when compared to previously used models of comparable size. However, a more careful analysis— including the types of control structures actually developed— is needed.

A network with 'state' vector of size 10, a 'memory' vector of size 20 and an additional hidden layer of 20 units was trained on a training set of 19% of all possible sentences with a single level of embedded PPs in either the subject or the object position, as well as all sentences without embedding, until the average bit error in the semantics vector was reduced to 1.2%. To keep training time and network size tractable, adjectival modifiers were omitted from the training set this time. As before, traces of unseen input sequences were examined to investigate the structural criteria the network uses for its task.

The main conclusion is that the network acquired the regularity that, when entering an embedding, the last NP seen moves to the first argument position in the embedded frame (in case it is not already the first argument in the matrix frame). This can be seen when testing how the network generalizes from the one-level embeddings seen during training to multi-level structures.

The network correctly generalized to two levels of embedding in sentence-final position, as shown in figure 5a. In this case, iteratively inserting the last NP into first argument position and rebuilding predicate and second argument produces the right results. After additional training on 1% of the sentences with 2-level sentence-final embeddings the network could handle structures of at least 5-levels.

The same principle is used for PPs embedded in the subject of the the main clause, as shown in Figure 5b. However, the second level of embedding wipes out the top-level subject in the first argument slot, which is never recovered after leaving the embedded clause (Figure 5c).

[6] Note that one element of the semantics that is omitted in this mixed spatio-temporal representation is the coreference relation between the head of an NP and the subject of an embedded relation. I.e., *a triangle below a circle touches a square* is analyzed essentially as 'a triangle is below a circle and a triangle touches a square'.

Additional training on precisely these problematic cases was unable to force the network to reinstate feature vectors from 'memory'. This confirmed that the network is not able to retain information which is not correlated with the output target for a number of intermediate processing steps, a limitation of SRNs previously found by Cleeremans et. al. [4].

5 Related work

A number of researchers have explored sequential recurrent networks (or related models) for language processing tasks.

Allen [1, 3, 2] uses networks were both the word input and a representation of the 'world' (corresponding to our semantic vectors) are given as inputs and the network typically has to give a yes/no answer regarding the correctness of the linguistic description.

Sopena [22] applies sequential networks to a domain very similar to ours (visual scenes), but his networks perform transductions of input to output *sequences* which closely parallel each other. The changing semantic representations are meant to model an attentional mechanism which shifts focus as the corresponding verbal sequences are processed.

St. John and McClelland's [23] network differs from ours mainly in the output representation. Instead of representing the full sentence semantics in parallel they use an additional 'selector' input to select the particular slot to be considered. Although the outputs are built incrementally like in our case almost all syntactic complexities are removed by treating entire phrases as input units and encoding sequential order information explicitly in the input vectors.

Several studies investigate the use of sequential networks for the recognition of scripts (i.e. schematic event sequences). Harris and Elman [13] use the word prediction training scheme to address the question how well SRNs can capture correlations between temporally distant script variables.

Miikkulainen and Dyer [19] use the same training target principle as adopted here, namely the network is required to transform a sequential information into a static structure. The issue of syntactic complexity is addressed by building in a two-level hierarchy, where output patterns of one level become the sequential inputs items at the next. Also of interest is the idea of inverting the SRN architecture for the purpose of sentence generation.

All authors except the first two explicitly want their networks to capitalize on priming effects due to statistical dependencies between elements in the semantic domain (e.g., certain relations tend to hold between certain objects and not others). The work reported here emphasizes the combinatorial nature of syntactic and semantic structure. As a result, such dependencies are mostly considered undesirable, except were they reflect syntactic regularities.

6 Conclusions

The aim of the work reported here was to explore the possibilities and limitations of simple recurrents network in a fairly natural language learning task, namely extracting semantic features from sequential word input.

Our first experiments with declarative sentences encoding single two-place predications and attributes showed that these networks can indeed learn to extract the appropriate correlations between word encodings, output features and sequential structure to incrementally assemble the complete semantics of such sentences. The network thus trained exhibited a fair amount of robustness in dealing with incomplete syntax, as well as sensitivity to fine points of sequential order.

However, our experiments with sentences containing embedded structures indicate that simple recurrent networks of the types investigated here are not able to retrieve information that was detected previously, but was temporarily uncorrelated with the target output during some phase in sequential processing. This led to an inability to correctly process center-embedded PPs, while multi-level embeddings in sentence-final position could even be generalized from one-level training cases.

Although this particular phenomenon may be in rough accordance with psycholinguistic data, a general inability to handle embedded structures is clearly not acceptable. Geoff Hinton (personal communication) has recently suggested applying full backpropagation in time [21] to this language learning problem, but it remains to be seen whether this is sufficient to learn to solve the center-embedding problem.

Simple recurrent networks have proven to be quite effective in extracting syntactic structure as it is manifest in *distributional* properties of the input. However, according to cognitively oriented linguistic theories [18, 17], the semantic and conceptual domain provides not only a target for language learning, but also constrains syntactic form to be motivated by the conceptual structure it conveys. For example, Langacker [18] suggests that there exists an iconic relationship between the syntactic unity of a noun phrase and the conceptualized unity of the properties it encodes (such as shape, size, color). The general idea that a combination of semantic grounding and distributional induction leads to syntactic structure is also consistent with theories of language acquisition and development [20].

Clearly the localist, flat feature encoding used here, although initially convenient and efficient for our purposes, does not have enough structure to explore any of those possibly crucial relationships between syntax and semantics. Even in our simple domain this lack became evident: there is no way the network could possibly generalize from the noun phrases it has seen in subject position to those in object position. The reason is that the relevant output features are completely disjoint and every pair of bits within one argument slot has no more in common than a pair of bits from different arguments.[7]

These considerations suggests that further progress towards learning of both natural language syntax and semantics will crucially depend on advances in the connectionist treatment of hierarchical representations, be they localist, structured models [8] or PDP mechanisms [14].

[7] Due to this second reason, the problem is not solved by simply replacing the local representation with a distributed one.

7 Acknowledgements

I wish to thank Jerry Feldman and Subutai Ahmad for many fruitful discussions, as well as comments on earlier versions of this paper.

References

[1] Robert B. Allen. Sequential connectionist networks for answering simple questions about a microworld. In COGSCI [5], pages 489–495.

[2] Robert B. Allen and Selma M. Kaufman. Identifying and discriminating temporal events with connectionist language users. In *IEEE Conference on Artificial Neural Networks*, London, October 1989.

[3] Robert B. Allen and Mark E. Riecken. Reference in connectionist language users. In *Connectionism in Perspective*, Zurich, October 1988.

[4] Axel Cleeremans, David Servan-Schreiber, and James L. McClelland. Finite state automata and simple recurrent networks. *Neural Computation*, 1(3):372–381, 1989.

[5] *Proceedings of the 10th Annual Conference of the Cognitive Science Society*, Montreal, Quebec, Canada, August 1988.

[6] *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, University of Michigan, Ann Arbor, Mich., August 1989.

[7] Garrison W. Cottrell and Fu-Sheng Tsung. Learning simple arithmetic procedures. In COGSCI [6], pages 58–65.

[8] J. Diederich. Instruction and higher-level learning in connectionist networks. *Cognitive Science*, 1(2):163–182, 1989.

[9] Jeffrey L. Elman. Finding structure in time. CRL Technical Report 8801, Center for Research in Language, University of California at San Diego, La Jolla, Calif., April 1988.

[10] Jeffrey L. Elman. Representation and structure in connectionist models. CRL Technical Report 8903, Center for Research in Language, University of California at San Diego, La Jolla, Calif., August 1989.

[11] Jeffrey L. Elman. Structured representations and connectionist models. In COGSCI [6], pages 17–25.

[12] Jerome A. Feldman, George Lakoff, Andreas Stolcke, and Susan Hollbach Weber. Miniature language acquisition: A touchstone for cognitive science. Technical Report TR-90-009, International Computer Science Institute, Berkeley, Calif., March 1990. Submitted to the 12th Annual Conference of the Cognitive Science Society.

[13] Catherine L. Harris and Jeffrey L. Elman. Representing variable information with simple recurrent networks. In COGSCI [6], pages 635–642.

[14] Geoffrey E. Hinton. Representing part-whole hierarchies in connectionist networks. In COGSCI [5], pages 48–54.

[15] M. I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, Amherst, Mass., August 1986.

[16] Ronald M. Kaplan and Joan Bresnan. Lexical functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, Mass., 1982.

[17] George Lakoff. *Women, Fire and Dangerous Things. What Categories Reveal about the Mind.* University of Chicago Press, Chicago, 1986.

[18] Ronald Langacker. *Foundations of Cognitive Grammar. Vol. 1: Theoretical Prerequisites.* Stanford University Press, Stanford, 1985.

[19] Risto Miikkulainen and Michael G. Dyer. A modular neural network architecture for sequential paraphrasing of script-based stories. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 49–56, Washington, D.C., June 1989.

[20] Steven Pinker. *Language Learnability and Language Development.* Harvard University Press, Cambridge, Mass., 1984.

[21] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1: *Foundations*, pages 318–362. Bradford Books (MIT Press), Cambridge, Mass., 1986.

[22] Josep Maria Sopena. Verbal description of visual blocks world using neural networks. Technical Report UB-DPB-88-10, Departament de Psicologia Basica, Universitat de Barcelona, Barcelona, Spain, December 1989.

[23] Mark F. St. John and James L. McClelland. Applying contextual constraints in sentence comprehension. In COGSCI [5], pages 26–32.

	Pred						Arg1								Arg2								
	Relation					Mod	Shape			Size			Shade		Shape			Size			Shade		
	T	L	R	A	B	F	C	S	T	S	M	L	D	L	C	S	T	S	M	L	D	L	
(a)	15	21	17	23	18	33	47	24	46	18	24	11	31	27	48	18	23	28	28	2	8	22	a
	3	28	22	24	20	37	97	0	5	3	20	0	2	5	39	26	34	24	26	1	3	23	circle
	74	8	1	12	7	0	99	0	5	5	0	0	2	2	27	22	39	12	12	0	0	14	touches
	98	1	1	4	1	0	99	0	0	1	0	1	5	0	26	32	32	8	8	0	0	6	a
	98	1	1	2	1	0	99	0	1	0	0	0	0	0	3	22	84	1	12	0	0	2	triangle
(b)	3	27	18	19	20	47	97	2	2	6	5	0	0	2	27	30	29	12	16	0	2	14	circle
	78	8	1	11	6	0	99	0	3	7	0	0	1	1	25	28	39	10	10	0	0	11	touches
	97	1	0	2	2	0	98	0	9	0	0	0	0	0	5	18	81	2	6	0	0	4	triangle
(c)	15	21	17	23	18	33	47	24	46	18	24	11	31	27	48	18	23	28	28	2	8	22	a
	3	28	22	24	20	37	97	0	5	3	20	0	2	5	39	26	34	24	26	1	3	23	circle
	4	26	17	26	19	31	99	0	0	2	12	1	4	2	14	44	41	9	18	0	1	10	a
	5	32	9	16	15	39	98	0	11	0	0	0	0	0	10	32	60	3	13	0	0	2	triangle
(d)	87	8	1	9	7	0	15	3	87	22	1	3	37	17	46	16	26	32	20	0	3	27	touches
	99	2	6	4	0	0	9	0	99	9	2	2	95	6	33	30	29	25	24	1	4	32	a
	99	3	6	3	0	0	0	0	99	0	0	0	99	0	17	22	58	7	36	1	2	26	triangle
(e)	15	21	17	23	18	33	47	24	46	18	24	11	31	27	48	18	23	28	28	2	8	22	a
	16	26	3	32	9	30	55	7	40	80	6	0	46	30	67	8	25	54	33	2	6	41	small
	13	25	13	25	20	28	67	2	62	98	1	0	99	9	77	20	8	69	44	9	52	51	dark
	16	32	21	24	18	13	17	0	99	98	0	0	99	0	57	15	29	38	45	14	45	40	triangle
	2	49	15	32	22	20	1	0	99	99	0	0	97	0	78	11	19	25	42	26	43	38	is ...
(f)	15	21	17	23	18	33	47	24	46	18	24	11	31	27	48	18	23	28	28	2	8	22	a
	21	23	21	20	17	32	47	8	58	7	28	3	95	26	46	31	25	36	46	3	17	38	dark
	28	21	3	26	9	30	71	0	80	35	6	0	99	9	81	5	16	64	48	6	19	65	small
	20	17	16	19	20	47	10	0	99	16	1	0	99	0	33	8	69	38	35	3	10	32	triangle
	3	30	16	22	37	48	0	0	99	0	1	0	98	0	61	15	21	26	33	1	8	11	is ...
(g)	15	21	17	23	18	33	47	24	46	18	24	11	31	27	48	18	23	28	28	2	8	22	a
	3	15	18	24	26	41	7	63	10	9	1	0	3	17	34	30	28	17	26	2	4	22	square
	2	17	27	21	32	41	5	96	3	2	8	3	1	10	21	41	27	12	14	0	1	10	is
	1	46	48	1	2	1	3	96	0	1	7	3	0	0	22	48	30	13	10	0	0	8	to
	2	59	45	2	0	0	2	97	0	5	3	3	1	0	20	39	31	12	9	0	0	7	the
	0	0	97	7	0	0	0	99	0	0	0	0	1	0	17	37	39	8	12	0	0	4	right
	0	0	98	2	0	0	0	99	0	0	0	1	0	0	14	38	39	6	13	0	0	4	of
	0	0	96	3	0	0	1	98	0	0	0	3	0	0	15	34	45	6	15	0	0	4	a
	0	0	95	1	0	2	0	86	1	0	1	0	0	0	2	8	94	4	30	0	0	2	triangle

Figure 3: Network performance on sample inputs. For each word processed the output layer activations are shown (multiplied by 100).

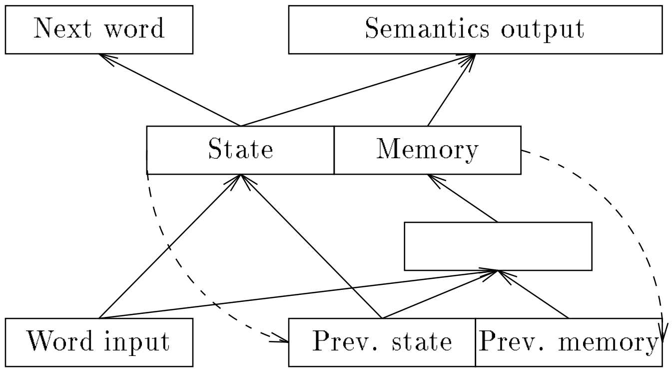

Figure 4: Extended recurrent architecture for merged predictive and semantics training.

	------Pred------						---------Arg1---------								---------Arg2---------								
	Relation					Mod	Shape			Size			Shade		Shape			Size			Shade		
	T	L	R	A	B	F	C	S	T	S	M	L	D	L	C	S	T	S	M	L	D	L	
(a)	13	4	11	19	28	0	19	6	67	0	0	0	0	0	31	47	24	0	0	0	0	0	a
	6	13	20	5	61	0	0	79	75	0	0	0	0	0	12	53	24	0	0	0	0	0	square
	99	0	3	4	0	0	0	99	2	0	0	0	0	0	18	48	29	0	0	0	0	0	touches
	98	0	1	7	2	0	0	99	0	0	0	0	0	0	15	49	36	0	0	0	0	0	a
	99	0	0	2	1	0	0	98	0	0	0	0	0	0	99	0	6	0	0	0	0	0	circle
	0	61	47	0	1	0	46	0	29	0	0	0	0	0	2	61	59	0	0	0	0	0	to
	0	78	14	0	0	0	73	0	38	0	0	0	0	0	21	38	42	0	0	0	0	0	the
	0	1	99	0	0	0	82	0	18	0	0	0	0	0	30	34	23	0	0	0	0	0	right
	0	0	99	0	0	0	99	0	1	0	0	0	0	0	28	49	24	0	0	0	0	0	of
	0	0	99	0	0	0	99	0	1	0	0	0	0	0	35	40	22	0	0	0	0	0	a
	0	0	99	0	0	0	99	0	0	0	0	0	0	0	0	99	0	0	0	0	0	0	square
	0	0	0	94	0	0	0	99	0	0	0	0	0	0	89	24	0	0	0	0	0	0	above
	0	2	0	98	8	0	0	99	0	0	0	0	0	0	94	7	1	0	0	0	0	0	circle
	0	0	1	98	1	0	0	99	0	0	0	0	0	0	2	2	98	0	0	0	0	0	triangle
(b)	13	4	11	19	28	0	19	6	67	0	0	0	0	0	31	47	24	0	0	0	0	0	a
	14	9	20	5	50	0	0	0	99	0	0	0	0	0	26	30	49	0	0	0	0	0	triangle
	0	1	1	98	0	0	0	0	99	0	0	0	0	0	42	18	35	0	0	0	0	0	above
	0	0	0	99	0	0	0	0	99	0	0	0	0	0	33	16	53	0	0	0	0	0	a
	0	1	0	99	0	0	0	0	99	0	0	0	0	0	0	99	0	0	0	0	0	0	square
	0	26	6	32	43	0	0	0	99	0	0	0	0	0	0	85	27	0	0	0	0	0	is
	0	36	59	0	0	0	0	0	100	0	0	0	0	0	4	87	9	0	0	0	0	0	to
	0	44	46	0	0	0	0	0	100	0	0	0	0	0	4	88	14	0	0	0	0	0	the
	0	99	0	0	0	0	0	0	100	0	0	0	0	0	0	98	3	0	0	0	0	0	left
	0	99	0	1	0	0	0	0	100	0	0	0	0	0	0	96	8	0	0	0	0	0	of
	0	99	0	3	0	0	0	0	99	0	0	0	0	0	1	98	1	0	0	0	0	0	a
	0	99	0	1	2	0	0	0	99	0	0	0	0	0	98	1	0	0	0	0	0	0	circle
(c)	13	4	11	19	28	0	19	6	67	0	0	0	0	0	31	47	24	0	0	0	0	0	a
	6	13	20	5	61	0	0	79	75	0	0	0	0	0	12	53	24	0	0	0	0	0	square
	0	0	0	99	0	0	0	99	0	0	0	0	0	0	30	33	31	0	0	0	0	0	above
	0	0	0	99	0	0	0	99	0	0	0	0	0	0	36	34	25	0	0	0	0	0	a
	0	1	0	99	0	0	0	99	0	0	0	0	0	0	99	0	0	0	0	0	0	0	circle
	0	0	0	1	98	0	52	0	8	0	0	0	0	0	51	26	17	0	0	0	0	0	below
	1	1	0	1	98	0	57	0	29	0	0	0	0	0	26	49	20	0	0	0	0	0	a
	0	0	2	0	98	0	99	0	0	0	0	0	0	0	0	1	99	0	0	0	0	0	triangle
	99	0	1	0	2	0	97	0	1	0	0	0	0	0	13	69	28	0	0	0	0	0	touches
	99	0	0	0	0	0	99	0	0	0	0	0	0	0	2	97	0	0	0	0	0	0	a
	96	0	0	0	6	0	99	0	0	0	0	0	0	0	99	4	0	0	0	0	0	0	square

Figure 5: Processing of embedded clauses.