Research Report AI-1993-01

Instantiating Thematic Roles with a Recurrent Neural Network

F. James Eisenhart

Artificial Intelligence Programs
The University of Georgia
Athens, Georgia 30602 U.S.A.

Instantiating Thematic Roles with a Recurrent Neural Network

F. James Eisenhart
Artificial Intelligence Programs
University of Georgia

Human languages have both a surface structure and a deep structure. The surface structure of a language can be described by grammatical rules which generate its well-formed sentences. Deep structure can be described by a set of thematic roles which specify the meaning of a sentence independent of any particular language. In this paper, I show that a recurrent neural network can learn to map English sentences generated by a unification-based grammar onto their appropriate thematic roles. Two models are evaluated: (1) a simple recurrent network like those of Jordan (1986) and Elman (1990), and (2) a modified recurrent network which incorporates a time-varying training signal, direct connections between its input and output units, and two separate state layers. It is found that the modified network learns faster and more completely than the simple recurrent network.

1. Introduction

Language learning involves mapping a variable surface structure onto an invariant deep structure. Every language has its own surface structure with a unique set of syntactic rules. Speakers of different languages make different use of cues like affixation and word order to interpret the meaning of a sentence. Speakers of American Sign Language even use cues in another modality, visuo-spatial rather than auditory-temporal, to interpret meaning (Bellugi, Poizner, and Klima 1989). Yet these different surface structures appear to be mapped onto the same deep structure. People are not born with a biological predisposition to speak a particular language. Instead, they learn whatever language they are raised with. This means that they learn to map the surface structure of a particular language onto the deep structure encoded in the human brain. The question is, how is this done?

One possibility is that a surface feature becomes associated with its deep meaning through a process of statistical inference. The idea is that the surface feature is heard repeatedly in a particular context. This context provides a semantic representation of the underlying event, some part of which is matched by the syntactic feature. A child who hears the two paired often enough eventually infers an association between them. Such associations add combinatorially so that new combinations of previously learned elements are still mapped onto the correct deep representation.

One problem with modeling this process is that we do not know how the brain represents the deep structure of language. It is possible to skirt this issue if we represent the deep structure of a sentence with a set of *thematic roles* like Agent, Action, and Theme, which correspond roughly to subject, verb, and object (Fillmore 1968). These roles are instantiated by the appropriate fillers from the sentence. For example, the sentence *The dog chased the policeman* has the thematic structure Agent = *dog*, Action = *chase*, and Theme = *policeman*. The thematic structure of a sentence is not dependent on its surface form, so *The policeman was chased by the dog* has the same thematic structure as the previous sentence.

The two connectionist models presented here learn to map English sentences generated by a unification-based grammar onto their appropriate thematic roles. The input sentences employ active and passive voice, prepositional clauses and several different verb forms. The models learn to map this input onto a semantic representation consisting of six different thematic roles.

2. Background

Most neural network models of sentence comprehension can be classified according to three overlapping criteria. The first is the type of representation used. Early models like Cottrell and Small (1983) and Waltz and Pollack (1985) used local representations, meaning that each unit represents a single, explicitly defined concept. Such models do not learn; instead, the relationships between concepts are encoded by hard-wired connections. Processing consists of a competitive spreading activation process, in which the network seeks a globally coherent state. More recent models use distributed representations for their hidden units, meaning that a concept is represented by a pattern of activation that is spread across all of the units. Distributed representations are usually created by the model itself via some learning procedure like backpropagation (Rumelhart, Hinton, and Williams 1986).

The second classification criterion is the model architecture. Early learning models like McClelland and Kawamoto (1986) used non-recurrent architectures in which a sequential input pattern (e.g., the words in a sentence) is represented non-sequentially across several sets of input units. Each set of input units represents a different point in time. This arrangement eliminates the need for recurrent connections to store context information, so the network can be trained with standard backpropagation. However, it also limits the length of an input sequence to the number of time-slices in the input layer. More recent models (St. John and McClelland 1990; Elman 1990) use some variation on a recurrent network architecture devised by Jordan (1986, see Figure 1a) which handles sequential inputs yet still works with standard backpropagation.

The final classification criterion is the task that the model preforms. Some models compute an explicit semantic representation of their input, as in thematic role instantiation (St. John and McClelland 1990; St. John 1992b). Others learn an internal semantic representation in response to some other task (Elman 1990; St. John 1992a). Within each of these categories, numerous semantic effects have been demonstrated, many by more than one model. These include lexical and structural disambiguation, instantiation of implicit constituents, anaphoric reference, and generalization to novel sentences.

The models presented here use distributed representations across their hidden units; they use recurrent networks; and they compute explicit semantic representations of their input.

3. The Task

The models' task is to use the surface structure of an English sentence to instantiate the appropriate thematic roles. They are presented with each word of the sentence across their input units, and they must produce the filler for each thematic role across their output units. The models learn this task from a series of examples in a training corpus. Their generalization ability is then tested by presenting them with a second corpus containing novel sentences and measuring their performance.

The thematic roles used are a subset of those of Parsons (1990) as clarified by Covington (1992). These roles and the rules that describe them simplify the actual situation in English in order to reduce the problem to a manageable size. They are as follows:

1. *Action*. Thing done. Verb.
2. *Agent*. Person or thing that causes an event to occur. Subject of an active sentence. Marked with *by* in a passive sentence.
3. *Theme*. Person or thing affected. Direct object of an active sentence. Subject of a passive sentence.
4. *Source*. Point of origin in an action involving a transfer. Indirect object. Marked by *from*.
5. *Goal*. Destination in an action involving a transfer. Indirect object in an active sentence. May be subject of a passive sentence if no Theme is present. Marked by *to*.
6. *Instrument*. Means by which an Action is accomplished. Object of *with*.

Each thematic role may appear in more than one surface position except for the Action, which is always the verb. Thematic roles are assigned to surface positions according to the following rules:

General Assumptions

1. An active sentence contains a past tense verb by itself; a passive sentence contains the

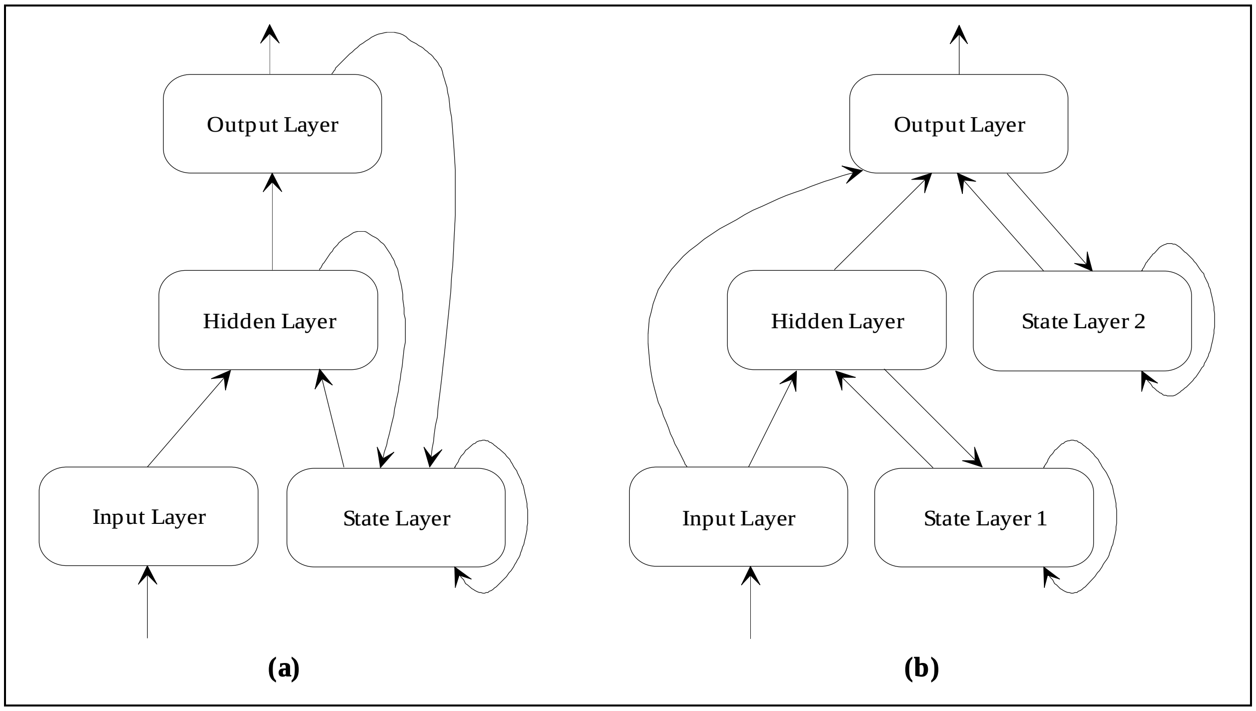

Figure 1. Two network architectures were used: *(a)* a simple recurrent network after Jordan (1986), and *(b)* a modified version with direct input-output connections and two state layers.

past participle preceded by *was*.
2. All sentences contain at least a subject and a verb.
3. Some sentences contain a direct object, which is a noun phrase following the verb, and/or one or more indirect objects, which are prepositional phrases or noun phrases following the verb.
4. Where they are used, the prepositional markers *by, from, to,* and *with* unambiguously mark the thematic role of their associated noun phrase.

Active Sentences
5. The Agent is the subject.
6. The Theme, if present, is the direct object.
7. The Goal, if present, may appear as a noun phrase immediately following the verb or as a prepositional phrase, as in *The thief gave the waitress the ring* and *The thief gave the ring to the waitress.*
8. Other constituents appear as prepositional phrases.

Passive Sentences
9. The Theme, if present, is the subject.
10. If a Goal, but no Theme, is present, then the Goal is the subject and the verb is followed by *to,* as in *The cat was talked to by the waitress.*
11. The Agent is marked with *by*.

The training corpus was generated by a Prolog and GULP (Covington 1992) program embodying the rules described above. A series of semantically appropriate thematic role instantiations were represented in a script (St. John and McClelland 1990). The possible fillers for each role were:

1. Action: *bark, chase, give, hit, kiss, meow, see, take, talk*
2. Agent: *cat, dog, policeman, thief, waitress*
3. Theme: *bone, cat, dog, mouse, policeman, ring, thief, waitress*
4. Source: *cat, dog, policeman, thief, waitress*
5. Goal: *cat, dog, policeman, thief, waitress*
6. Instrument: *billy_club, fist, purse*

The program generated all possible events and surface forms from the script, creating a total of 192 sentences. From these, 15 were chosen at random to be the generalization corpus. These were set aside and not presented to the network during training. The remaining 177 sentences constituted the training corpus.

4. Model 1: A Simple Recurrent Network[1]

[1] The models were created with the PlaNet neural network tool (Miyata 1991).

4.1 Network Architecture and Processing. The first model is a simple recurrent network (SRN) like those of Jordan (1986) and Elman (1990) (see Figure 1a). It has 29 input units, 35 hidden units, 35 output units, and 64 state units. Each input unit represents a single word. At the start of each sentence, the network is reset by setting all of the units' activations to 0. On each input cycle, the input unit for the current word is set to 1. Activation then spreads forward through the hidden units to the output units. Each output unit represents a particular thematic role/filler combination. At the end of a sentence, the filler unit for each instantiated role should be on and all other units should be off. Processing continues for as many cycles as there are words in the sentence.

The state units encode contextual information from previous input cycles. There is one state unit for each hidden or output unit. Each hidden or output unit feeds back onto its corresponding state unit, and each state unit feeds back onto itself, giving it a kind of temporal memory. Each state unit is activated according to the function

$$activation = activation * decay + recinput$$

where *decay* is a parameter that determines the rate at which the memory trace stored in the state units decays (the value 0.6 was used) and *recinput* is the recurrent input from the corresponding hidden or output unit.

The activation of the hidden and output units is determined by the function

$$activation = logistic(netinput + bias)$$

where *netinput* is the sum of the activation times the connecting weight for each unit connected to the hidden or output unit

$$netinput_i = \sum_{all\ connected\ i} activation_j * weight_{i,j},$$

bias is a positive or negative offset for each unit, and *logistic* is the standard non-linear activation function

$$logistic(x) = \frac{1}{1 + e^{-x}}.$$

The model was trained by backpropagation (Rumelhart, Hinton, and Williams 1986) with a learning rate of 0.2 and a momentum of 0.9. On each learning epoch, the sentences in the training corpus were presented in a pseudo-random order. The role/filler targets were matched to the network outputs and the weights adjusted after each word.

4.2. Results. The network was trained for 1,000 cycles. Its learning performance was

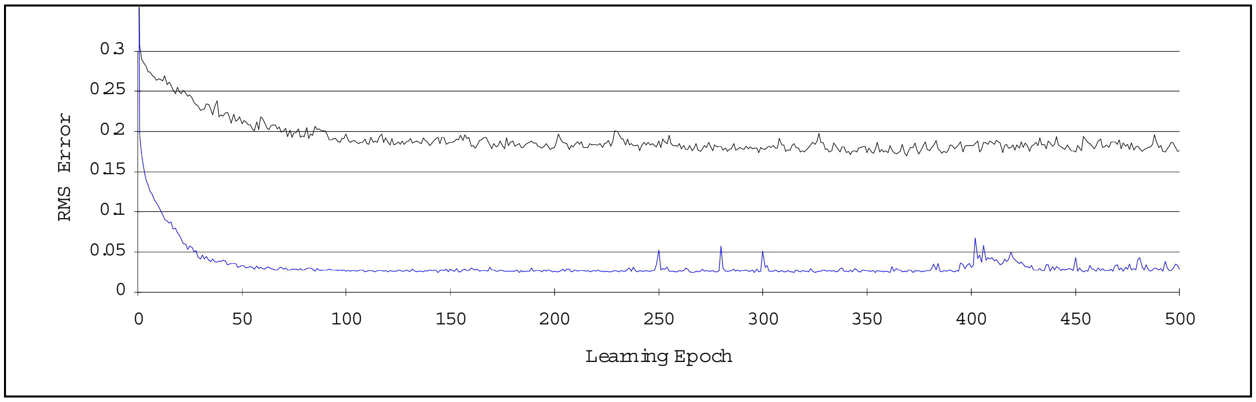

Figure 2. Root-mean-squared (RMS) error as a function of learning epoch. The top line is for the simple recurrent network and the bottom is for the modified network.

measured in three ways. First, the root-mean-squared error (RMS) for each learning epoch was recorded. The RMS error is the square root of the average across all of the output units of the square of the difference between their activation and their training target:

$$RMS = \sqrt{\frac{1}{N_{all}} \sum_{outputs} (target - output)^2},$$

where *N* is the number of outputs. The RMS error for each learning epoch is the average error over all of the patterns in the training corpus. As shown in Figure 2, the RMS error declined during the first 100 epochs, indicating that the network was learning, but it stagnated at about 0.18 thereafter. It did not improve when training was extended to as many as 2,000 epochs.

Second, the model was tested every 50 epochs for instantiation errors. An instantiation error is the failure to activate a role/filler unit that should be on or the activation of a role/filler pair that should be off. Instantiation errors were measured by setting an activation floor of 0.3, below which any unit was considered to be off, and an activation ceiling of 0.7, above which any unit was considered to be on. Any unit which should be on but was below 0.7 or should be off but was above 0.3 was considered to be in error. The error for each sentence was measured by counting the number of output units in error after the entire sentence had been presented. These errors were then tabulated for each learning epoch. As shown in Figure 3, the model's performance improved dramatically for the first 100 epochs, slightly for the next 100, and little thereafter. At its lowest RMS error (epoch 950, not shown) the model still made 81 instantiation errors, an average of 0.46 per sentence. Again, its performance did not improve when training was extended to as many as 2,000 epochs.

Third, the model was tested for its generalization performance using the weights from epoch 950. The generalization corpus, consisting of 15 sentences not seen during training, was presented to the model once, without learning. The resulting RMS error was 0.1836, and the model made 14 instantiation errors, an average of 0.93 per sentence.

4.3 Discussion. Overall, the model learned its task marginally well. It learned to instantiate most roles correctly. Its average instantiation error rate of 0.46 per sentence is equivalent to about

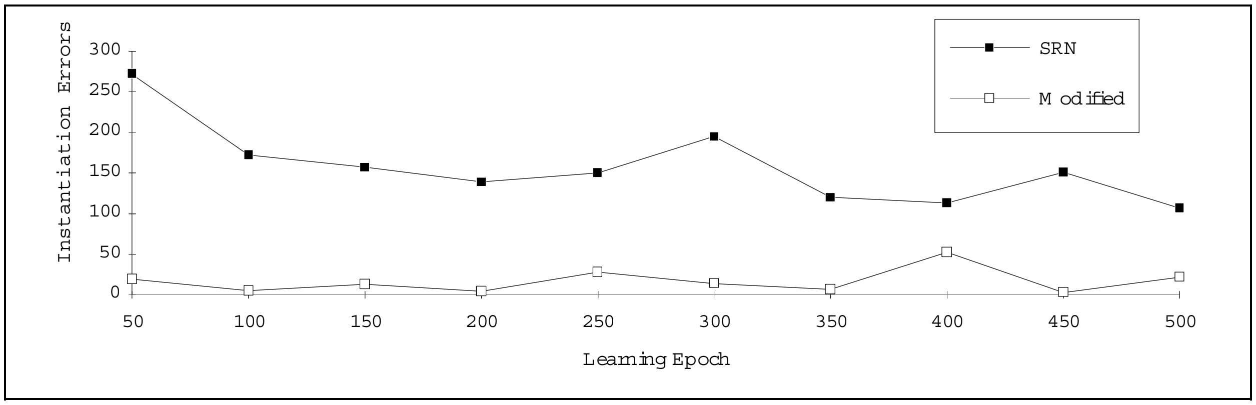

Figure 3. Role instantiation errors as a function of learning epoch for the simple recurrent network (top) and the modified network (bottom). Errors were measured every 50 epochs.

one error for every seven instantiations. However, its generalization performance was much worse (one error for every three to four instantiations), and it showed no signs of improvement out to 2,000 epochs.

There seem to be two reasons for this. The first concerns the training signal used. Because the same training signal, containing all of the role/filler pairs, was presented after each word of a sentence, the network had to predict all of the instantiations after each word. Obviously, in a sentence like *The cat kissed the waitress* it does not have enough information to predict the fillers for any of the roles after the first word, and it may not be able to say who the cat kissed until the last word. Consequently, it will always have some level of residual error that it cannot eliminate. This residual error hampers the model's ability to learn the task because it is much larger than the non-residual error. Thus, it may swamp out the meaningful error signals that would reduce the number of instantiation errors.

The second reason is that the SRN architecture is not ideal for learning a grammar. To learn a grammar, the network must learn a large number of identity mappings between words and their meanings (St. John 1992b). For instance, *dog* might mean Agent=*dog* or Theme=*dog*, but it cannot mean Agent=*cat*. The network could encode these mappings most efficiently as a set of disjunctive relationships between its input and output units. This is not possible in an SRN because there are no direct connections between the inputs and outputs. Instead, all activation must go through a limited number of hidden units.

5. Model 2: A Modified Recurrent Network

5.1 Network Architecture and Processing. This model differs from the previous one in several ways. First, it uses a time-varying training signal in which the role/filler pair for each thematic role is not set to 1 until its identifying word appears in the sentence. For example, in *The cat chased the dog* the Agent=*cat* target would be set to 1 beginning with cycle 2, but the Theme=*dog* target would not be set to 1 until cycle 5.

Second, the network includes direct connections between its input and output units (see Figure 1b). These allow it to efficiently learn identity mappings between words and their meanings.

Third, the single state layer of Model 1 is split into separate layers for the hidden and output units in Model 2. This forces the hidden units to concentrate on encoding patterns of surface features because they no longer have access to the state of the output units. Without output information encoded in the hidden units, the network relies more heavily on the direct input-output connections, which improves generalization.

Fourth, because the hidden units only code for surface features in this model, fewer of them are needed. Only 10 hidden units were used in the simulations presented here.

5.2 Results. The model was trained for 500 epochs, producing the RMS error values shown in Figure 2. The RMS error declined to about 0.03 within 50 epochs and changed little thereafter. The network also produced few instantiation errors within 50 epochs (see Figure 3) and did not change greatly thereafter. Generalization was again tested by presenting the network with the generalization corpus using the weights from the learning epoch with the lowest RMS error (epoch 450). This produced an RMS error of 0.0331 and 5 instantiation errors (0.33 per sentence).

5.3 Discussion. As shown in Table 1, this model learned much better than the first. Its learning curve was steeper than Model 1's and it reached its best RMS error in half as much time. RMS errors for learning and generalization were both much lower than for Model 1, although this is in part due to the lower error inherent in Model 2's time-varying training signal. More revealingly, Model 2 had far fewer instantiation errors than Model 1 for both learning and generalization. The low numbers on these measures indicate that Model 2 learned the training corpus almost perfectly and in such a way that could generalize to new examples.

Table 1. Comparative results for Models 1 and 2.

Test	Model 1	Model 2
Epochs to Learning Plateau	100	50
Lowest RMS Error	0.1633	0.0217
Fewest Instantiation Errors	107	3
Generalization RMS Error	0.1836	0.0331
Generalization Instantiation Errors	14	5

6. Conclusions

These results illustrate two important principles of neural network modeling. First, a supervised network cannot learn a task unless the information that it needs is present in its training signal. In this case, Model 1 had to predict the deep structure of a sentence before it had seen the surface information that it needed. In such a case, a network will minimize its error by generating output values between its conflicting targets, but it will not learn the mapping. In Model 2 this problem was solved by giving the network a more learnable training signal.

Second, the architecture of a neural network model can have a tremendous effect on its performance. In this case, revising the SRN architecture of Model 1 by adding direct input-output connections and a second state layer allowed Model 2 to encode identity mappings between its input and output units more concisely. This shortened the model's learning time and improved its generalization performance.

References

Bellugi, Ursula, Howard Poizner, and Edward S. Klima (1989). Language, modality and the brain. *Trends in Neurosciences* **12**:380-388.

Cottrell, Garrison W., and Steven L. Small (1983). A connectionist scheme for modeling word sense disambiguation. *Cognitive Science* **6**:89-120.

Covington, Michael A. (1992). *Natural language processing for Prolog programmers.* Publication draft. To be published by Prentice-Hall, 1993.

Elman, Jeffrey L. (1990). Finding structure in time. *Cognitive Science* **14**:179-211.

Fillmore, Charles J. (1968). The case for case. In *Universals in linguistic theory*, ed. E. Bach and R. T. Harms, 1-88. New York: Holt, Rinehart and Winston.

Jordan, Michael I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the eighth annual conference of the Cognitive Science Society*, 531-546. Amherst, MA.

McClelland, James L., and A. H. Kawamoto (1986). Mechanisms of sentence processing: Assigning roles to constituents of sentences. In *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 2: Psychological and biological models*, by J. L. McClelland, D. E. Rumelhart, and the PDP Research Group, 272-325. Cambridge: MIT Press.

Miyata, Yoshiro (1991). A user's guide to PlaNet version 5.6: A tool for constructing, running, and looking into a PDP network. University of Colorado, Boulder.

Parsons, Terence (1990). *Events in the semantics of English: A study in subatomic semantics.* Cambridge: MIT Press.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, by J. L. McClelland, D. E. Rumelhart, and the PDP Research Group, 318-364. Cambridge: MIT Press.

St. John, Mark F. (1992a). Learning language in the service of a task. *Proceedings of the fourteenth annual conference of the Cognitive Science Society*, 271-276. Hillsdale, NJ: Erlbaum.

St. John, Mark F. (1992b). The story gestalt: A model of knowledge-intensive processes in text comprehension. *Cognitive Science* **16**:271-306.

St. John, Mark F., and James L. McClelland (1990). Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence* **46**:217-257.

Waltz, David L. (1985). Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science* **9**:51-74.