To the Graduate Council:

I am submitting herewith a thesis written by Fermin Soriano Salkjelsvik entitled "Natural Language Processing with Recurrent Neural Networks." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Bruce J. MacLennan, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Vice Provost
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under the rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Fermín Soriano Schlyelowich

Date November 30, 1989

NATURAL LANGUAGE PROCESSING WITH RECURRENT NEURAL NETWORKS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Fermin Soriano Salkjelsvik

December 1989

To Elizabeth and Bobby

ABSTRACT

Various experiments have been done on natural language using neural networks. Qualities such as distributed representation, the ability to generalize and efficient parallel computation make neural networks very attractive for natural language processing. In this work, a particular model is considered: recurrent neural networks. Their main advantage over feed-forward models is that they require fewer units to represent the input, and this results in a smaller network. Their main disadvantage is that they are harder to train. Some solutions to this problem are presented along with a comparison in performance of recurrent neural networks versus feed-forward neural networks in three different tasks: syntactic feature extraction, sentence comparison and translation between English and Spanish.

ACKNOWLEDGEMENTS

I want to thank my major professor, Dr. Bruce J. MacLennan, for introducing me to neural networks and for all the guidance that I have received throughout this project. His criticism and positive attitude were essential to the completion of this work. I also would like to thank the other members of the committee, Dr. David Mutchler, who assisted me with good suggestions and criticism, and Dr. Bethany Dumas, who introduced me via several seminars to a different and more exciting approach to natural language processing.

I would like to give a very special thanks to Bobby England for helping me get to Knoxville, for his friendship throughout my course of study, and for his cooking.

Finally, I would like to thank my wife, Elizabeth, for all her love and understanding. She assisted me not only as a wonderful companion but also as a colleague, taking the time to read and critique my work.

TABLE OF CONTENTS

		Page
1. Introduction		1
2. Neural Networks Applied to Natural Language Processing		4
2.1	Basic Form	4
2.2	Recurrent Neural Networks	7
2.3	Current Applications in Natural Language Processing	8
3. Experiments with Recurrent and Feed-Forward Network Models		11
3.1	Terms, Abbreviations and Notation	11
3.2	Feature Extraction	12
3.2.1	Model description	13
3.2.2	Training session	15
3.2.3	Generalization	15
3.3	Sentence Comparison	20
3.3.1	Model description	21
3.3.2	Training session	23
3.3.3	Generalization	23
3.4	Translating Between English and Spanish	29
3.4.1	Model description	29
3.4.2	Training session	32
3.4.3	Generalization	33
3.5	Sequential Recurrent Network Training Methods	39
4. Experimental Results and Conclusions		43
4.1	Overall Performance	43
4.2	Generalization	44
4.3	Scaling	45
5. Future Work		47
Notes		49
Bibliography		50

APPENDICES 53

Appendix A: Examples of Training Sets 54

Appendix B: Network Definition Files 57

Vita 60

LIST OF TABLES

Page

Table 1 - Feature Extraction Using 80% of the Sentences as the Learning Set 16

Table 2 - Feature Extraction Using 60% of the Sentences as the Learning Set 16

Table 3 - Feature Extraction Using 40% of the Sentences as the Learning Set 17

Table 4 - Feature Extraction Using 20% of the Sentences as the Learning Set 17

Table 5 - Feature Extraction: Variance in Number of Epochs and TSS 18

Table 6 - Sentence Comparison Using 80% of the Sentences as the Learning Set 25

Table 7 - Sentence Comparison Using 60% of the Sentences as the Learning Set 25

Table 8 - Sentence Comparison Using 40% of the Sentences as the Learning Set 26

Table 9 - Sentence Comparison Using 20% of the Sentences as the Learning Set 26

Table 10 - Sentence Comparison: Variance in Number of Epochs and TSS 27

Table 11 - Translation between English and Spanish Using 80% of the Sentences as the Learning Set 34

Table 12 - Translation between English and Spanish Using 60% of the Sentences as the Learning Set 34

Table 13 - Translation between English and Spanish Using 40% of the Sentences as the Learning Set 35

Table 14 - Translation between English and Spanish Using 20% of the Sentences as the Learning Set 35

Table 15 - Translation between English and Spanish: Variance in Number of Epochs and TSS 36

LIST OF FIGURES

	Page
Figure 1 - General Flow	6
Figure 2 - A Sequential Recurrent Network	8
Figure 3 - Model 1: Feed-Forward Network for Feature Extraction	14
Figure 4 - Model 2: Sequential Recurrent Network for Feature Extraction	14
Figure 5 - Feature Extraction: Percentage of Incorrect Features Using 80%, 60%, 40% and 20% of the Sentences as Learning Sets	19
Figure 6 - Feature Extraction: Percentage of Incorrect Sentences Using 80%, 60%, 40% and 20% of the Sentences as Learning Sets	19
Figure 7 - Model 1: Feed-Forward Network for Sentence Comparison	22
Figure 8 - Model 2: Sequential Recurrent Network for Sentence Comparison	22
Figure 9 - Percentage of Incorrect Comparisons for Models 1 and 2	27
Figure 10 - Model 1: Feed-Forward Network to Translate between English and Spanish	30
Figure 11 - Model 2: Sequential Recurrent Network to Translate between English and Spanish	31
Figure 12 - Percentage of Incorrect Translated Words for Models 1 and 2	38
Figure 13 - Percentage of Incorrect Translated Sentences for Models 1 and 2	38
Figure 14 - Step 1: Learning Using a Parameter μ Equal to 0	41
Figure 15 - Step 2: Learning Using a Parameter μ Equal to 0.6	41

CHAPTER 1

INTRODUCTION

Since the very early days of computers, people have wanted to process natural language with computer systems. Their final goal was to create a system capable of understanding natural language. That is, they wanted to be able to give commands and queries in natural language and have the machine both understand them and also reply in natural language. This would allow man-machine communication without the need of technical expertise. But early work in natural language processing turned out to be more difficult than originally expected:

> Expectations were very high. This led to a booming research activity as early as the late fifties.... The early expectations proved to be overoptimistic and unrealistic. This led to disillusionment - if not to say disappointment - and to a negative overreaction.[1]

Some of the problems encountered were due not only to the fact that natural language is highly irregular, but also to our inability to extract and represent the meaning of a sentence in a feasible way. Research almost came to a stop. Natural language processing became a theoretical topic -- the desire to understand how humans can handle such a complex system as natural language -- with little hope of practical application. Recently, however, research in natural language processing has gained new momentum. One important reason is that researchers have learned from the past and have more realistic expectations. Another reason is the advancement in both computational linguistics and Artificial Intelligence techniques, which have provided some solutions to language analysis and representation. Also, research in natural language processing is becoming more important in fields such as Linguistics, Psychology and Computer Science. Some ambitious, long-term projects have been launched; important examples are the Heidelberg Science Center program by IBM Corporation in

Germany, and the activities of the Japanese Fifth Generation Computer Project, in which natural language processing is a major highlight. This interest in language processing has already resulted in applications for storage and retrieval of documents, support in text processing (e.g. hyphenation, spelling aid, style and grammar checking), assistance in translation tasks, and man-machine communication.

Connectionism is an approach to knowledge representation which differs from other Artificial Intelligence representations such as scripts, frames and schemata. In connectionist models of computation, information processing takes place through the interactions of simple processing elements called units, which send excitatory or inhibitory signals to other units. The knowledge is represented in the connections and the strength of those connections, and it is distributed throughout the network.

Connectionist learning procedures provide new insights into natural language processing. The connectionist approach has some inherent properties that are advantageous for natural language processing and that are difficult to achieve with traditional logic-based models. [25] These properties are:

* the ability to compute a nearest match instead of requiring an exact match,
* learning,
* fault-tolerance through the integration of overlapping modules, and
* the possibility of scaling up connectionist models to handle larger problems.

In traditional Artificial Intelligence there are basically two approaches to the representation of word meaning: formalistic and episodic. The formalist approach assumes a componential view in which a word's meaning is represented by a set of semantic features. In the episodic

approach, on the other hand, the meaning of the word must be defined in terms of its relationship to other words and available memory structures. Interestingly enough, there are connectionist models that are inspired by both views. [11]

Connectionist models have in some cases demonstrated behavior similar to children in dealing with language tasks. [19] Given this and the fact that very simple connectionist networks can do certain non-trivial language processing tasks, there is some reason to believe that connectionist models bring us closer to understanding or at least imitating how the human brain handles natural language.

Neural networks are connectionist models inspired by biological processes. They offer an alternative type of computing (analog) to the traditional Von-Neumann model (digital), on which most of today's computers are based. Although neural networks were first developed in the 1940's, the interest in them have in the 1980's been stronger than ever. [5]

CHAPTER 2

NEURAL NETWORKS APPLIED TO NATURAL LANGUAGE PROCESSING

In the following sections, the origin and the basic form of artificial neural networks are covered. A special model, recurrent neural networks, will be introduced, along with a description of current applications of neural network models in natural language processing.

2.1 Basic Form

Neural network models are inspired by biological models. The interest in neural networks is prompted by the fact that even very small animals can solve problems that are very difficult for conventional computers. In the 1940's, McCulloch and Pitts designed a model inspired by the behavior of neural systems.[2] This model had interesting computational properties and could be used to solve simple pattern matching problems. Later research, however, established limitations to the kind of functions that this model could learn, and the interest dwindled thereafter.

In the 1980's, interest and research in neural networks re-emerged due primarily to:

* advances in mathematical theories,
* the development of new computer tools, and
* an increased understanding of neurobiology.

An artificial neural network consists of simple, neuron-like processing elements called "units" that interact via weighted connections. Each unit has a "state" or "activation level" that is determined by the input received from other units. The net input to a particular unit i, for example, is simply the sum of the influences of all of the other units in the network. The influence of some other unit j, for example, is the product of that unit j's output times the weight of the connection to unit i. Thus, the net input to unit i is given by the following formula:

$$\text{net input}_i = \sum_j (\text{activation}_j * \text{weights}_{ij})$$

To find the activation of a unit, a function is applied to the net input for that unit. The activation function is normally one of the following four types.

* Linear: the activation of a unit is simply equal to the net input.
* Linear Threshold: the activation of a unit is normally 1 or 0, depending on whether the net input is larger than the threshold value.
* Stochastic: the activation of a unit is computed by applying a logistic function to the net input. The main difference here is that the update rule specifies only a probability that the units will take on one or the other of their values. This kind of activation function is used in the Boltzmann machines.
* Continuous Sigmoid: the activation of a unit is non-linearly related to its net input according to the logistic function:

$$\text{activation}_i = \frac{1}{1 + e^{-\text{net input}}}$$

The computing process in a neural network can be stated as the summation of weighted inputs across the net to produce an output. For each unit, the activation is calculated by applying some function to the net input. The general flow of information is described in figure 1.

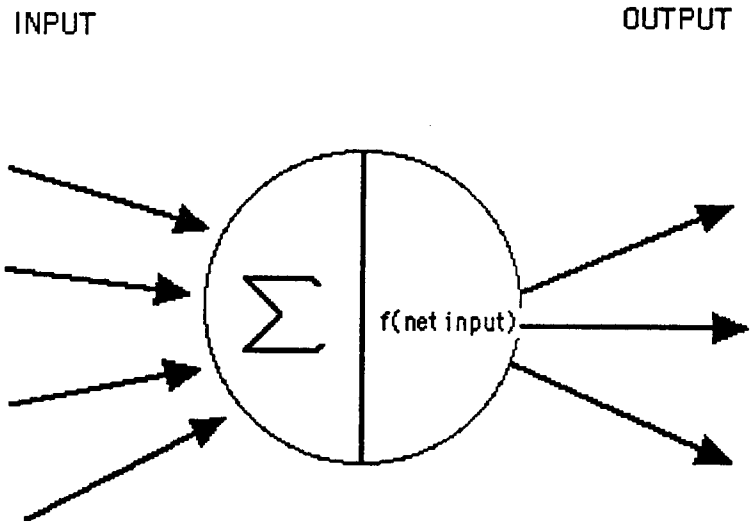


Figure 1

General Flow

Short-term knowledge in a connectionist network is encoded by the states of the units. Long-term knowledge, on the other hand, is encoded by the connections and by their weights, so learning consists of changing the weights or adding or removing connections. [8] Learning in a neural network typically occurs after the activations of the output units have been computed. The resulting activations of the output units are compared to the target output values to yield an error for that particular input pattern. Learning involves reducing that error by modifying the weights so that the error function approaches a minimum. Networks containing hidden

units are harder to train, since the learning procedure must decide what the hidden units should represent. Learning procedures can be divided up into three main categories:

* supervised procedures, which require a teacher to provide the correct output,
* reinforcement procedures, which require a single scalar evaluation of the output, and
* unsupervised procedures, which construct internal models to capture the correct input/output mapping.

2.2 Recurrent Neural Networks

Recurrent neural networks are special neural network models whose design include some feed-back connections. These feed-back connections allow recurrent networks to compute complex time-dependent calculations. Recurrent networks were first studied in detail by Hopfield and Tank who demonstrated how such networks could be used to solve combinatorial optimization problems such as the travelling salesman problem. [7]

Different recurrent models have been applied succesfully in different fields. A kind of recurrent networks called "Cellular Automata" are used in physics for their ability to simulate physical phenomena such as fluid flow. [7] In 1986, Jordan applied a sequential recurrent network to language processing in what he called a "Connectionist Sequential Machine". [10]

The first training algorithm used for recurrent neural networks was based on "simulated annealing" and was used in a recurrent network with symmetric connections called the "Boltzmann Machine". Back propagation and other training algorithms have also been successfully applied to recurrent networks. Improved learning algorithms, however, need to be

found since current techniques do not work very well for networks with many units. [5]

The recurrent model used in this paper is based on a sequential recurrent network (see figure 2). In this model, feed-forward processing occurs as described in section 2.1, Basic form, creating different states in the network. The state of the units in the hidden layer are copied back to the so called "memory units" or "current state units" for the next feed-forward sweep. More specifically, the activation of a memory unit is set equal to the copy from the hidden layer plus μ (mu) times the unit's previous activation. This is done to help the memory units capture the history of previous inputs in the network.

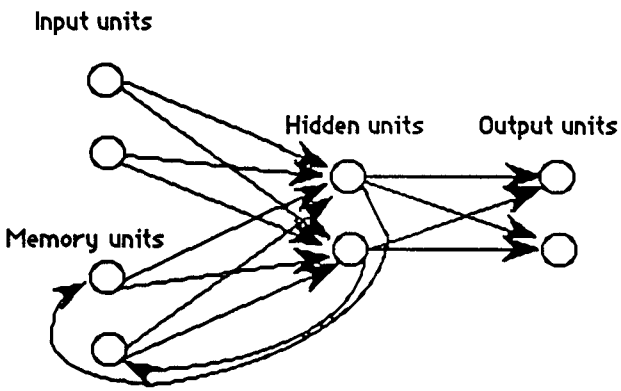


Figure 2

A Sequential Recurrent Network

2.3 Current Applications in Natural Language Processing

Recent experiments using connectionist models have been successful in completing some language processing tasks. In most cases, these experiments have been done on domains

requiring a relatively small set of input and output units. Even so, the results achieved by these models open the door for a completely new approach to language processing.

One well known experiment is the NETtalk grapheme-to-phoneme system by Sejnowsky and Rosenberg (1987), a net that learns to "read aloud" by mapping a string of English letters to a string of English phonemes. A neural network with one hidden layer was used to perform this task. The network processes the text sequentially in seven letter chunks. The pronunciation of the middle letter is encoded by 26 output units that determine the phoneme and other features such as stress and syllable boundaries. After extensive training, the network showed a good ability to generalize correctly on new examples.

In 1986, Rumelhart and McClelland developed a verb learning network, which transforms a phonetic representation of the root form of a verb into the phonetic representation of the past tense of the verb. The model produced correct output for novel verbs and handled special cases such as "eat/ate", or "sing/sang". One very interesting part of the experiment was comparing the results they obtained from the network to those obtained from children who were learning the past tenses of English verbs.

> We have shown that a reasonable account of the acquisition of past tense can be provided without recourse to the notion of a "rule" as anything more than a description of the language .... The children need not figure out what the rules are, nor even that there are rules.[3]

Another aspect of language often studied is speech recognition. It is very difficult to define a set of rules to determine which sound is being produced and separate different sounds in continuous speech. This task becomes even more difficult when we consider that the way a sentence is said varies according to a person's mood or intention and is normally different from one time to another. Then, when we try to recognize speech from different speakers, the problem is further complicated. A rule-based system would have to cover a broad spectrum

of possibilities. On the other hand, neural networks learn through the training process what aspects of the input patterns are important; this should make neural networks insensitive to a person's mood or voice, since they are, in principle, capable of finding the relationship among different input patterns that produce the same output.

There are a variety of studies exploring the capabilities of neural networks in language processing. Connectionists models are now being designed to translate sentences into case-frame representations (McClelland and Kawamoto 1986), index causal chains for narrative recall (Golden 1986), store and retrieve relational data (Hinton 1986), solve some prepositional phrase attachment problems (Touretzky 1989), paraphrase script-based stories (Miikkulainen and Dyer 1989), do pronoun reference and translation (Allen 1987) and assemble words of a sentence (Nolfi and Parisi 1988).

In time, we should be able to decide which language processing tasks are appropriate for neural networks and which are not. The results shown by the experiments mentioned above give us some reason for optimism. This optimism is shared by James L. McClelland:

> I believe that we are now on the verge of understanding sequential processes in connectionist networks, and that this will soon make it possible to describe a complete connectionist mechanism for language processing that captures both the strengths and limitations of human language capabilities.[4]

CHAPTER 3

EXPERIMENTS WITH RECURRENT AND FEED-FORWARD NETWORK MODELS

In this section, the notation and terms used in the experiments are defined. This is followed by a comparison of the performance of recurrent and feed-forward network models on three different language processing tasks: feature extraction, sentence comparison and translation between English and Spanish.

3.1 Terms, Abbreviations and Notation

input pattern - *n* bit vector that encodes the input sentence, where *n* is the number of bits needed to encode the sentence.

output pattern - *n* bit vector that encodes the output produced by the network, where *n* is the number of bits needed to encode the output pattern.

target pattern - *n* bit vector that represents the correct output pattern, where *n* is the number of bits needed to encode the output pattern.

input units - units that receive the input pattern from a source outside the network.

output units - units whose activations represent the output pattern.

hidden units - units that are neither input or output units. They receive their input from the input units or other hidden units.

memory units - units that contain feedback connections and whose activations represent the prior history of activation in the network.

learning set - set of input and target patterns used during the training of the network.

novel set - set of input and target patterns that have not been used during the training of the network.

epoch - training cycle where all the patterns in the learning set are presented and the weights adjusted.

pattern sum of squares (PSS) - the squared error terms summed over all of the output units.

total sum of squares (TSS) - the cumulative sum of the PSS for all patterns thus far processed within the current epoch.

learning rate - in the back propagation algorithm, the learning involves a change of the weight proportional to the weight error derivative. The learning rate is the constant of proportionality.

momentum - parameter that determines the effect of past weight changes on the current direction of movement in the weight space.

μ (mu).- parameter that determines the effect of past activations on the memory units.

3.2 Feature Extraction

The purpose of this feature extraction experiment was to test the ability of recurrent and feed-forward network models to deal with a simple language processing task. For a given sentence, each network had to extract some syntactic information about that sentence. This information is referred to as features, and for each sentence there are a total of six features, as follows:

* whether the sentence was in active or passive voice,
* whether the subject contained an article (e.g. 'the') or not,
* whether the object contained an article (e.g. 'the') or not,
* whether the order of the sentence was normal (Subject-Verb-Object) or not,

* whether the sentence was of the type Subject-Verb (no object) or not, and
* whether the sentence contained a preposition (e.g. 'to') or not.

3.2.1 Model description

A total of 269 sentences were produced using 15 different input words to serve as input patterns. The sentences were generated by making all possible combinations of subject-verb-object, subject-verb and object-subject-verb in both active and passive voice. Two neural network models were implemented: model 1 and model 2, described below.

Model 1 is based on a feed-forward architecture where an entire sentence is processed in one step (see figure 3). Since the longest input sentence is 8 words, 8 word slots are provided to encode the input sentence. Each word slot consists of 15 units to represent the word present at that particular slot. The input layer consists therefore of 120 units and is fully connected to a hidden layer of 20 units. The hidden layer is then fully connected to the output layer, which uses 6 units to represent the six output features. The total number of connections for model 1 is 2520.

Model 2 is based on a sequential recurrent network architecture where a sentence is processed one word at the time (see figure 4). Since there are 15 input words, we only need 15 units for the input layer. In order to keep track of the history of previous activations, 20 memory units are provided. The input units and the memory units are fully connected to a hidden layer of 20 units. For each unit in the hidden layer there is a feedback connection to an associated memory unit. The hidden layer is fully connected to the output layer consisting of 6 output

units. The number of connections for this model is 840, which makes this model three times smaller than model 1.

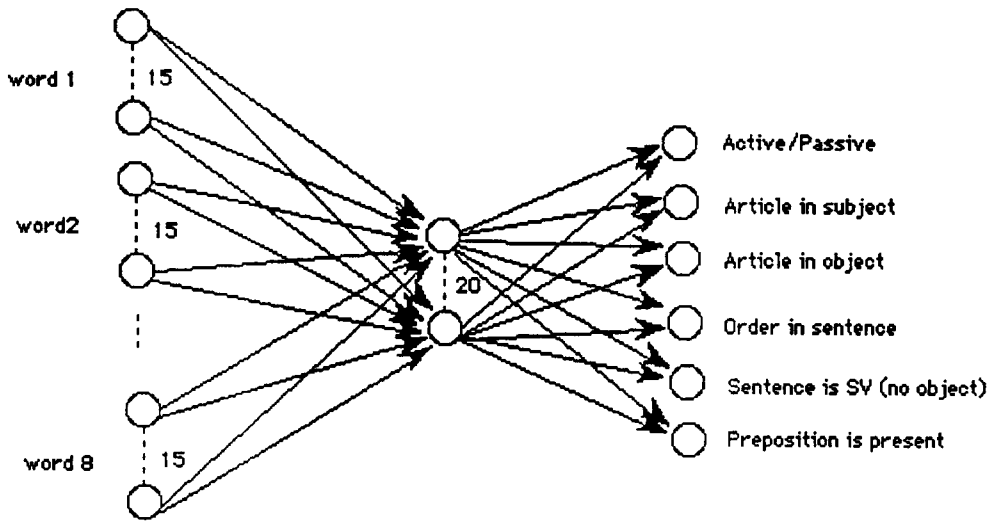


Figure 3

Model 1: Feed-Forward Network for Feature Extraction

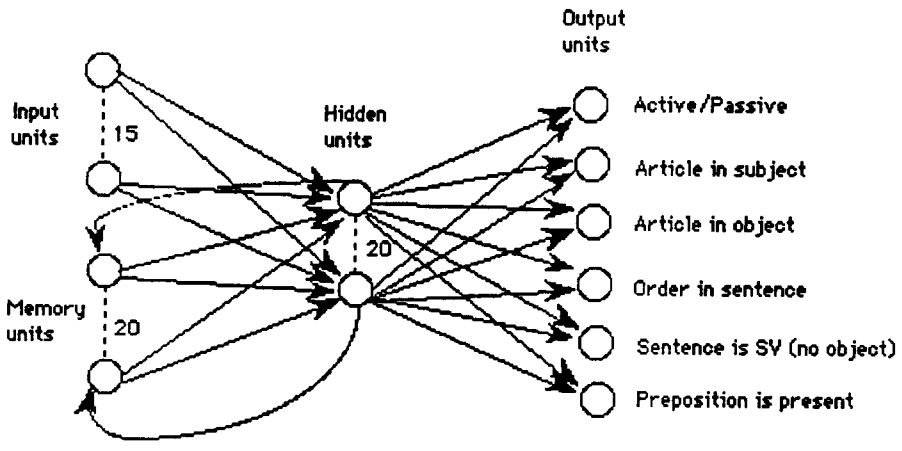


Figure 4

Model 2: Sequential Recurrent Network for Feature Extraction

3.2.2 Training session

A momentum of 0.5 and a learning rate of 0.2 were used for both models in all the feature extraction experiments except for one case. A momentum of 0.3 and a learning rate of 0.1 were used for model 2 when 20% of the sentences were used as the learning set. The smaller momentum and smaller learning rate were needed to slow the learning process and avoid convergence on local minima. The values for the momentum and the learning rate were chosen through trial and error; the selected values converged in most cases towards a global minimum in the error function within a small number of epochs. In these experiments, training stopped when the TSS fell below 0.3. For the sequential recurrent model, a parameter μ equal to 0 was used. Since the activation of a memory unit is set equal to the copy of the activation from the hidden layer plus μ times the unit's previous activation, μ equal to 0 means that an exact copy of the first hidden layer is made without taking into account the previous activation states in the memory layer.

The number of input patterns was approximately six times smaller for model 1 than for model 2 since model 1 represented the whole sentence in one input pattern, while model 2 required a different input pattern for every word in the sentence.

3.2.3 Generalization

Both models were tested for their ability to generalize correctly on patterns not seen previously. Four experiments were conducted teaching both networks with 80%, 60%, 40% and 20% of the sentences and then testing for generalization with 20% of the sentences not used during their training process. Each experiment was repeated with 3 different random samples; the results

reflect the average of the three samples (see tables 1, 2, 3 and 4). Table 5 presents the variance in the samples. The sentences for the learning set were generated and selected by a program that used a pseudo-random number generator (see Appendix A for examples). Different seeds for the pseudo-random number generator were used to create new samples.

For both models, an activation above 0.5 in an output unit was treated as a 1, meaning that the syntactic feature was present in the input sentence. Accordingly, an activation below or equal to 0.5 was treated as 0, and the syntactic feature was considered to be absent.

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Features	Percentage of wrong Sentences
Model 1: Feed-Forward	126.7	215	0.18	0.0	0.0
Model 2: Recurrent	210.7	1283	0.24	0.0	0.0

Table 1
Feature Extraction Using 80 % of the Sentences as the Learning Set

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Features	Percentage of Wrong Sentences
Model 1: Feed-Forward	134.30	161	2.04	0.93	3.70
Model 2: Recurrent	213.67	965	0.87	0.31	1.92

Table 2
Feature Extraction Using 60 % of the Sentences as the Learning Set

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Features	Percentage of Wrong Sentences
Model 1: Feed-Forward	144.67	108	3.50	1.13	3.70
Model 2: Recurrent	294.67	637	3.50	1.13	6.17

Table 3
Feature Extraction Using 40 % of the Sentences as the Learning Set

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Features	Percentage of Wrong Sentences
Model 1: Feed-Forward	137.0	54	13.56	5.45	22.84
Model 2: Recurrent	516.33	321	29.99	11.52	41.36

Table 4
Feature Extraction Using 20 % of the Sentences as the Learning Set

	Variance in Number of Epochs		Variance in Total Sum of Squares	
Experiment	Model 1 (Feed-Forward)	Model 2 (Recurrent)	Model 1 (Feed-Forward)	Model 2 (Recurrent)
80%	3.30	23.17	0.02	0.12
60%	2.49	53.27	1.61	0.58
40%	5.44	38.42	1.88	1.05
20%	9.27	140.30	0.69	8.98

Table 5
Feature Extraction: Variance in Number of Epochs and TSS

The two networks generalized perfectly after training them with 80% of the sentences. Using 60% and 40% of the sentences as learning sets, both models generalized well, with under 1.2% of the features wrong (see figure 5). Even with 20% of the sentences used as learning set, the percentage of incorrect features for the novel sentences is relatively low -- 5.4% for model 1 and 11.5% for model 2.

Over 94% of the sentences are classified correctly when 80%, 60% or 40% of the sentences are presented in the learning process. However, with 20% of the sentences used as the learning set, model 1 classified incorrectly 22.84% of the sentences while model 2 classified incorrectly 41.36% of the sentences (see figure 6).

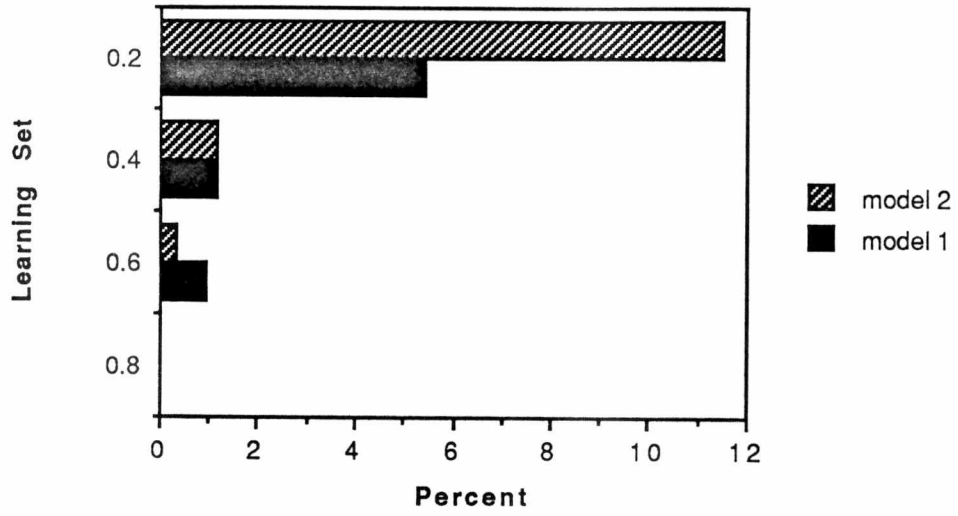


Figure 5

Feature Extraction: Percentage of Incorrect Features Using 80%, 60%, 40% and 20% of the Sentences as Learning Sets

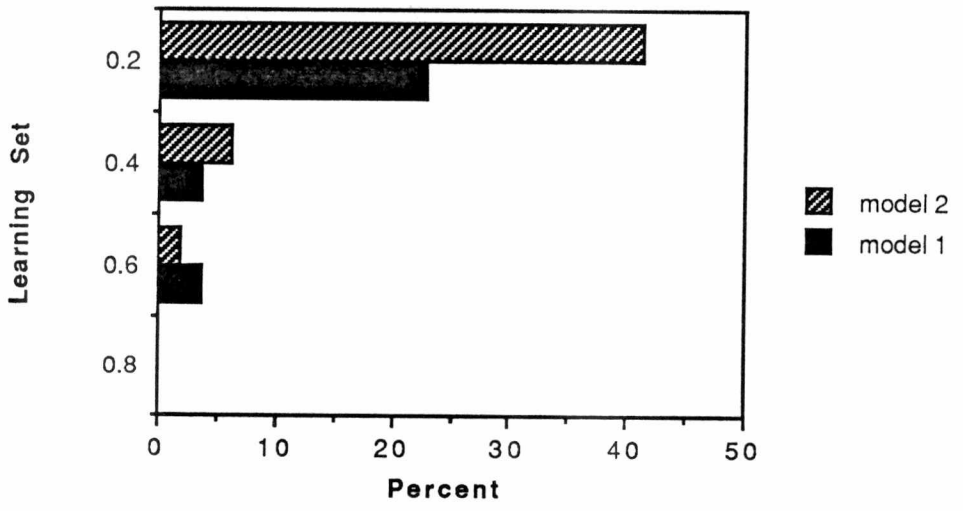


Figure 6

Feature Extraction: Percentage of Incorrect Sentences Using 80%, 60%, 40% and 20% of the Sentences as Learning Sets

Both models were able to perform a natural language task such as feature extraction with a limited number of words. Overall, both networks showed a good ability to generalize on novel inputs and the recurrent model performed as well as or better than the feed-forward model when presented with 40% or more of the sentences during the learning process. At the same time, the number of connections for the recurrent model was three times smaller than the feed-forward model.

It is important to remark that the network to begin with did not have any knowledge on how to solve the problem. The network was presented bit strings of input and the corresponding correct bit strings of output. There was no other information to help the network to learn what was going on. The network had to figure out what common characteristics of different input patterns produced similar output patterns; it did not know what the input bits stood for nor what the output bits represented; everything had to be learned by experience.

3.3 Sentence Comparison

The previous work done in extracting features from a sentence showed the ability of recurrent neural networks to extract some features by taking into account not only the words, but also the order in which the words appear. In this experiment, two neural networks models are implemented to learn how to decide on the basis of surface structure alone whether two sentences have the same meaning or not. The sentences used in this experiment were in passive and active voice, and in some cases, the word order in the sentence was altered.

3.3.1 Model description

A total of 400 pairs of sentences were produced using a total of 10 words to serve as input patterns. The sentences were generated by making all possible combinations of subject-verb-object and object-subject-verb in both active and passive voice. Again, two neural network models were implemented: model 1 and model 2.

In Model 1, based on a feed-forward architecture, both sentences are entered at the same time in one step (see figure 7). Since the longest sentence is 8 words long, 8 word slots are provided for each of the two sentences. Each word slot consists of 10 units to represent the word present at that particular slot. The input layer consists therefore of 160 units, and they are fully connected to a hidden layer of 30 units. This hidden layer is fully connected to a second hidden layer of 7 units. The second hidden layer is fully connected to the single output unit. There is a total of 5017 connections in model 1.

In Model 2, which is based on a sequential recurrent architecture, both sentences are fed in parallel into the network one word at the time (see figure 8). There are 10 input units for each sentence and two memory layers to compress the incoming sentences.

The input units for the first sentence and the first memory layer are connected to a hidden layer of 10 units, and, similarly, the input and memory units for the second sentence are connected to 10 units in the hidden layer. The two memory layers receive feed-back connections from the corresponding hidden layer. Both hidden layers are fully connected to another hidden layer of 5 units, and this hidden layer is fully connected to the output unit. The number of connections for model 2 is 525.

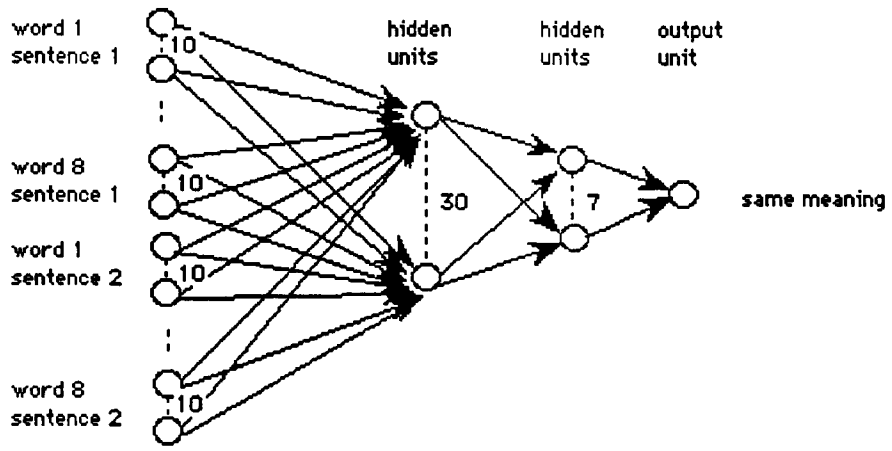


Figure 7

Model 1: Feed-Forward Network for Sentence Comparison

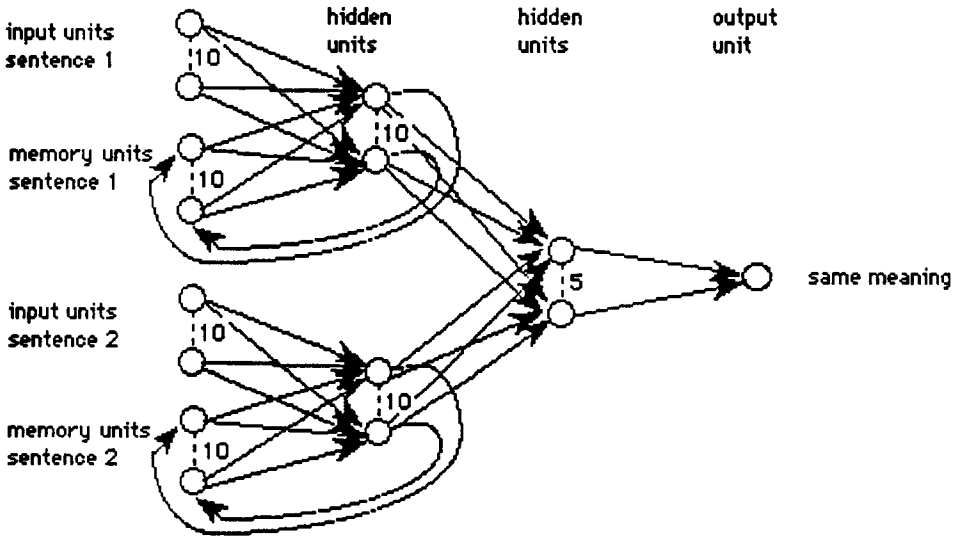


Figure 8

Model 2: Sequential Recurrent Network for Sentence Comparison

3.3.2 Training session

To train the network described in model 1, a momentum of 0.5 and a learning rate of 0.2 were used. For model 2, a momentum of 0.5 and a learning rate of 0.2 were used on the first 500 epochs while a momentum of 0.3 and a learning rate of 0.1 were used on the remaining epochs. A parameter μ equal to 0 was used for the memory units in the recurrent model. Since the activation of a memory unit is set equal to the copy of the activation from the hidden layer plus μ times the unit's previous activation, μ equal to 0 means that an exact copy of the first hidden layer is made without taking into account the previous activation states in the memory layer. The values for the momentum and the learning rate were chosen through trial and error; the selected values converged in most cases towards a global minimum in the error function within a small number of epochs. Learning stopped when the TSS value fell below 0.3, but in the experiments done using model 2, the TSS did not fall below 0.3, so training stopped after 3000 epochs. In all the cases, model 2 was able to learn correctly over 97% of the sentences in the learning set.

The number of input patterns was approximately six times smaller for model 1 than for model 2, since in model 2 the two sentences are entered one word at the time, thus requiring a new input pattern for each word entered. Due to the size of the input file for model 2, the training process using model 2 was very slow.

3.3.3 Generalization

Both models were tested for their ability to generalize to patterns not seen previously. Four experiments were conducted teaching both networks with 80%, 60%, 40% and 20% of the pairs

of sentences and then testing for generalization with 20% of the pairs of sentences not used during their training process. Each experiment was repeated with 2 different random samples; the results reflect the average of the two samples (see tables 6, 7, 8 and 9). The variance for these samples is reported in table 10. The pairs of sentences for each learning set were selected by a program that used a pseudo-random number generator (see Appendix A for examples). Different seeds for the pseudo-random number generator were used to create new samples.

For both models, an activation above 0.5 in the output unit was treated as a 1, indicating that the input sentences mean the same. Conversely, an activation below or equal to 0.5 was treated as 0, indicating the sentences have different meanings.

Neither model generalized correctly after learning with 80% of the sentences. Model 1 was incorrect in over 15% of the comparisons while model 2 was incorrect in 10% of the comparisons. Model 1 generalized correctly in 74% of the cases when 40% or more of the sentences were used as learning sets. Model 2, on the other hand, generalized correctly on almost 85% of the cases when 40% or more of the sentences were used as learning sets. Model 2 also generalized better than model 1 when 20% of the sentences were used during the learning process, model 2 missing on 23% of the comparisons while model 1 missed in almost 35% of the cases (see figure 9).

Overall, both models generalized worse than in the feature extraction task. The recurrent model had some difficulties in the learning process but showed a good ability to generalize, doing it correctly on almost 85% of the cases when the network was taught with 40% or more of the sentences. It is worth noticing that the recurrent network model required almost ten

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Sentences
Model 1: Feed-Forward	444.5	320	11.55	16.25
Model 2: Recurrent	3000	2270	6.98	9.38

Table 6
Sentence Comparison Using 80% of the Sentences as the Learning Set

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Sentences
Model 1: Feed-Forward	343.5	240	15.34	21.87
Model 2: Recurrent	3000	1706	8.56	13.13

Table 7
Sentence Comparison Using 60% of the Sentences as the Learning Set

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Sentences
Model 1: Feed-Forward	637	160	19.74	27.5
Model 2: Recurrent	3000	1130	11.19	15.63

Table 8
Sentence Comparison Using 40% of the Sentences as the Learning Set

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Sentences
Model 1: Feed-Forward	183.5	80	23.41	34.37
Model 2: Recurrent	3000	568	16.5	22.5

Table 9
Sentence Comparison Using 20% of the Sentences as the Learning Set

	Variance in Number of Epochs		Variance in Total Sum of Squares	
Experiment	Model 1 (Feed-Forward)	Model 2 (Recurrent)	Model 1 (Feed-Forward)	Model 2 (Recurrent)
80%	67.5	0.0	2.54	2.89
60%	30.5	0.0	2.36	0.77
40%	148.0	0.0	4.84	0.87
20%	44.5	0.0	3.125	2.21

Table 10
Sentence Comparison: Variance in Number of Epochs and TSS

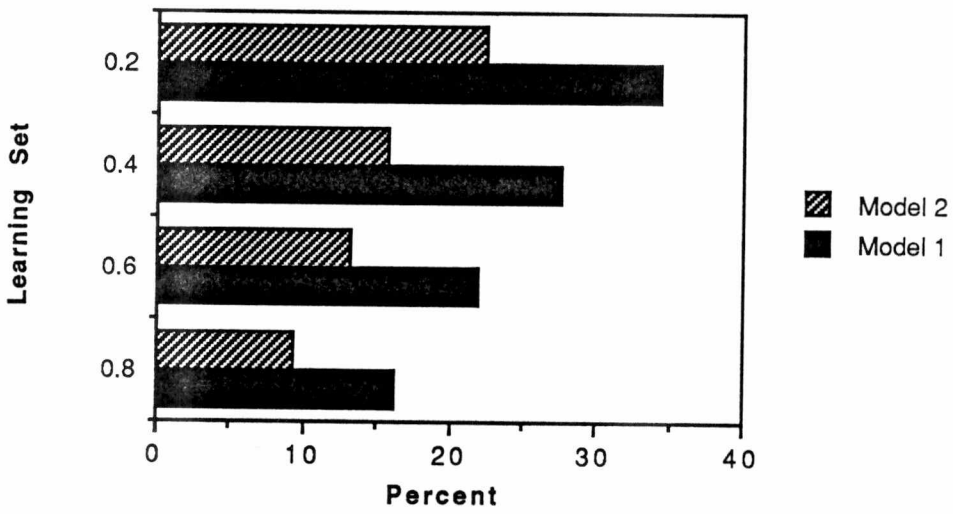


Figure 9

Percentage of Incorrect Comparisons for Models 1 and 2

times fewer connections than the feed-forward model, and that the overall results seem to favor the recurrent approach.

The "60% overlap test" is an alternative, very simple method to estimate whether two sentences mean the same or not. This method works as follows: if the number of words in common between the two sentences amounts to 60% or more of the number of words in the longer sentence, then the two sentences are estimated to mean the same. Likewise, if fewer than 60% of the words are identical, the sentences are estimated to have different meaning. This method was applied to the 400 pairs of sentences and was wrong in 32.5% of the cases.

The performance of the "60% overlap test" is worse than the sequential recurrent network approach, and it is only better than the feed-forward network approach when the network is trained on 20% or less of the sentences. Even with 40% of the sentences used as the learning set, the performance of the recurrent network is twice as good as the "60% overlap test".

In order to improve the performance of neural networks on sentence comparison additional information about the two sentences could be used as part of the input. That information could be provided by another neural network, such as the syntactic feature extraction in the previous experiment, or from another source, such as using the "60% overlap test" as an indication on the similarity between the two sentences.

Notice that the architecture for the sequential recurrent model involves the use of two different memory layers to encode the two input sentences instead of using one single memory layer. Two memory layers were used because they proved to learn better than when using a single memory unit.

3.4 Translating Between English and Spanish

In this series of experiments, feed-forward and recurrent network models learned to translate simple sentences from English to Spanish. In doing so, both networks had to infer not only the equivalence among English and Spanish words, but also the differences in word order, gender and prepositions between the two languages. This translation required no semantic analysis and no syntactic analysis in the traditional sense.

3.4.1 Model description

A total of 9 English words were used to produce 162 sentences. The sentences were generated by making all possible combinations of subject-verb-object in active voice. The corresponding Spanish translation required a total of 13 Spanish words. Additional Spanish words were needed because Spanish articles and adjectives have two different forms depending on the gender of the noun they modify. Two models were implemented in this experiment: model 1 and model 2, described below.

Model 1 is based on a feed-forward architecture where the translation process occurs in one step. The English sentence is fed into the input units, and the activation is propagated throughout the network producing the corresponding Spanish sentence in the output layer (see figure 10). In order to encode the English sentences, 8 word slots are provided in the input layer since the longest English sentence used is 8 words long. Each slot consists of 9 units to represent the word present at that particular word slot. The input layer consists therefore of 72 input units. These input units are fully connected to a hidden layer of 20 units. The hidden units are at the same time fully connected to the output units. To encode the output

sentence, model 1 requires 117 units, since the longest Spanish sentence is 9 words and since 13 different Spanish words were required in the translation. The total number of connections for model 1 is 3780.

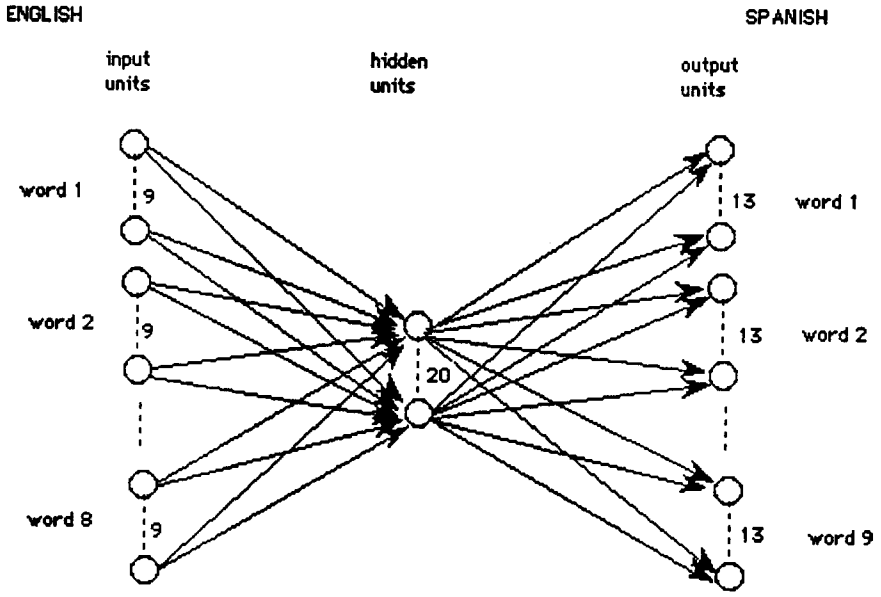


Figure 10

Model 1: Feed-Forward Network to Translate between English and Spanish

Model 2 is based on a sequential recurrent architecture where the English input sentence is entered one word at a time and the Spanish output sentence is likewise produced one word at a time (see figure 11). In order to do this, the recurrent model needs an end of sentence marker to indicate the end of both the input and the output sentence. A total of 10 input units are used to encode the 9 English words and the end of sentence marker. The end of sentence marker is necessary for the network to be able to determine when the end of the input and the output sentences occurred. In order to keep track of the previous input words

and their order in the input sentence, a memory layer of 40 units is provided. The input units and the memory units are fully connected to a hidden layer of 40 units. For each unit in the hidden layer there is a feedback connection to a corresponding memory unit. Since the output sentence is also produced one word at the time, we need another memory layer of 14 units to record the output words already produced in that sentence. Each unit in the second memory layer has a feed-back connection from an associated output unit. This memory layer and the first hidden layer are fully connected to another hidden layer of 40 units. This second hidden layer is also fully connected to the output layer consisting of 14 output units, 13 of which are used to encode a Spanish word while the remaining unit indicates the end of the output sentence. The number of connections for model 2 is 4734. This large number of connections is primarily due to the number of memory units needed by the recurrent model to represent the translation process.

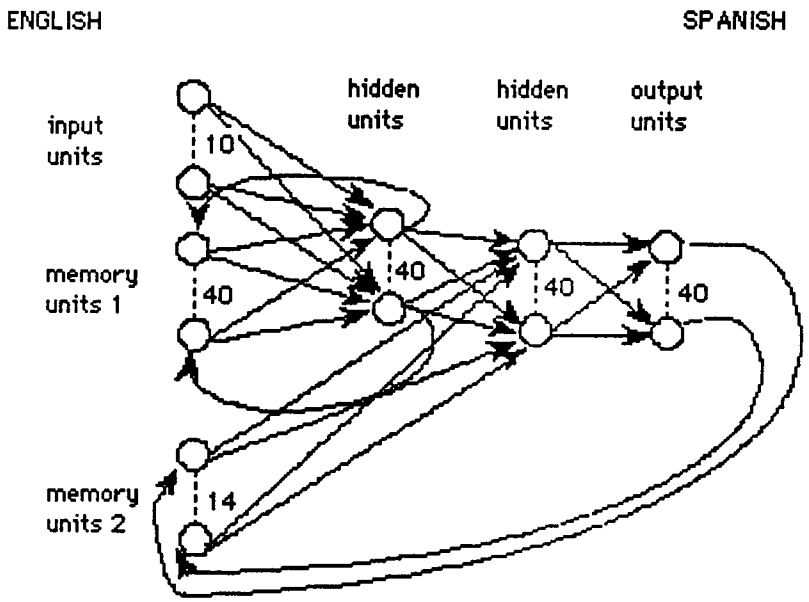


Figure 11

Model 2: Sequential Recurrent Network to Translate between English and Spanish

3.4.2 Training session

A momentum of 0.5 and a learning rate of 0.2 were used for model 1, while a momentum of 0.3 and a learning rate of 0.1 were used for model 2. The values for the momentum and the learning rate were chosen through trial and error; the selected values converged in most cases towards a global minimum in the error function within a small number of epochs. A smaller momentum was required for the recurrent model in order to avoid oscillation in the learning process. Oscillation in the learning process occurred when the TSS did not decrease constantly but rather alternated between increasing and decreasing values.

The number of input patterns was approximately 13 times smaller for model 1 than for model 2 since model 1 represented the whole sentence in one input pattern, while model 2 required a different input pattern for every word in the English sentence and for every word in the corresponding Spanish sentence. The training of the recurrent model was divided into two steps, since it proved to be a very difficult task for this kind of network to learn. First, the network was trained to find the right weights in the connections between the first hidden layer and the combined initial layer of input units and first memory units. The main purpose of this step was to let the network represent each input sentence with a different state in the first memory layer. In other words, the network learned to distinguish among different sentences by encoding them as different states in the first memory layer. In the second step, the network was taught to produce the output sentence one word at a time given the encoded input sentence as the activation of the first memory layer. This method is described in more detail in section 3.5, Sequential Recurrent Network Training Methods.

Language translation requires much more than a simple mapping among words of two different languages. Notice that the network had to learn the following irregularities:

Gender: the article 'the' in Spanish can be 'el' or 'la', depending on the gender of the noun. The same applies to the adjective 'white', it can be either 'blanco' or 'blanca'.

Word order: adjectives in Spanish normally appear after the noun whereas in English they appear before the noun. For example, 'the white cat' is translated as 'el gato blanco'.

Adding a preposition: the verb 'to see' in Spanish ('ver') requires the preposition 'a' when it is followed by a person, but not when the verb is followed by a non-human. For example, 'the cat sees john' would be translated as 'el gato ve a juan', while 'john sees the cat' would be translated as 'juan ve el gato'.

Contracting preposition + article: The Spanish preposition 'a' followed by the article 'el' forms the contracted form 'al'.

3.4.3 Generalization

To check how well both models generalize, the two networks were taught with 80% of the sentences and then tested for generalization on the remaining 20%. Learning stopped when a TSS below 0.7 was reached. The experiment was repeated teaching model 1 with 60%, 40% and 20% of the sentences. Each experiment was repeated with 3 different random samples; the results reflect the average of average out of the three samples (see tables 11, 12, 13 and 14). The variance in the samples is reported in table 15. The pairs of sentences for each learning set were selected by a program that used a pseudo-random number generator (see Appendix A for examples). Different seeds for the pseudo-random number generator were used to create new samples.

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Words	Percentage of Wrong Sentences
Model 1: Feed-Forward	615	130	4.60	2.06	11.47
Model 2: Recurrent	4000	2150	138.58	39.65	69.78

Table 11
Translation between English and Spanish Using 80 % of the Sentences as the Learning Set

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Words	Percentage of Wrong Sentences
Model 1: Feed-Forward	835.67	97	11.10	5.73	22.92

Table 12
Translation between English and Spanish Using 60% of the Sentences as the Learning Set

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Words	Percentage of Wrong Sentences
Model 1: Feed-Forward	761	65	23.97	9.99	41.67

Table 13
Translation between English and Spanish Using 40% of the Sentences as the Learning Set

	Number of Epochs	Number of Patterns	TSS on Novel Set	Percentage of Wrong Words	Percentage of Wrong Sentences
Model 1: Feed-Forward	535.0	32	72.90	26.87	87.5

Table 14
Translation between English and Spanish Using 20% of the Sentences as the Learning Set

	Variance in Number of Epochs		Variance in Total Sum of Squares	
Experiment	Model 1 (Feed-Forward)	Model 2 (Recurrent)	Model 1 (Feed-Forward)	Model 2 (Recurrent)
80%	0.82	0.0	2.35	17.83
60%	25.94	-	2.35	-
40%	34.07	-	8.70	-
20%	23.85	-	13.75	-

Table 15
Translation between English and Spanish:
Variance in Number of Epochs and TSS

For both models, an activation above 0.5 in an output unit was treated as a 1, meaning that that the word was present in the generated Spanish sentence. An activation below or equal to 0.5 was treated as 0, indicating the word was not present.

Using 80% of the English sentences as the learning set, model 1 learned the correct translation for all the sentences of the learning set, while model 2 was able to learn the correct translation for all but 3 words in the learning set. Model 1 generalized correctly for over 88% of the novel sentences while model 2 generalized and translated correctly only 30% of the sentences. Considering the number of words to be translated, model 1 translated correctly almost 98% of the words while model 2 translated 60% of the words correctly. Since model 2 did not generalize well when presented with 80% of the sentences, experiments using 60%, 40% and 20% of the sentences were not done with this model.

When presented with 60% of the sentences during the learning process, model 1 translated correctly 77% of the novel sentences, missing on average 5.7% of the words (see figure 12). Using 40% of the sentences as the learning set, model 1 stills performs correctly on 58% of the novel sentences, missing only 10% of the words. Model 1's performance when trained with 20% of the sentences is markedly worse; the network translated correctly only 12% of the sentences missing on average 27% of the words (see figure 13).

Considering that the translation process was non-trivial, it is remarkable that a recurrent network (model 2) was to some degree able to encode a sentence entered one word at a time and then produce the corresponding translation also one word at a time. However, the recurrent network seemed to need many memory units to encode the translation process resulting in an even larger network than the feed-forward model. On the other hand, it is interesting that the feed-forward model (model 1) was able to translate correctly over 90% of the words in novel sentences when the network had been trained with only 40% or more of the sentences.

In order to improve the performance of both neural networks in the translation task, certain information about the translation process could be provided as part of the input. The two neural network models used had to learn everything involved in the translation process by experience. Since the translation task is fairly complex, a large learning set is require to generalize properly.

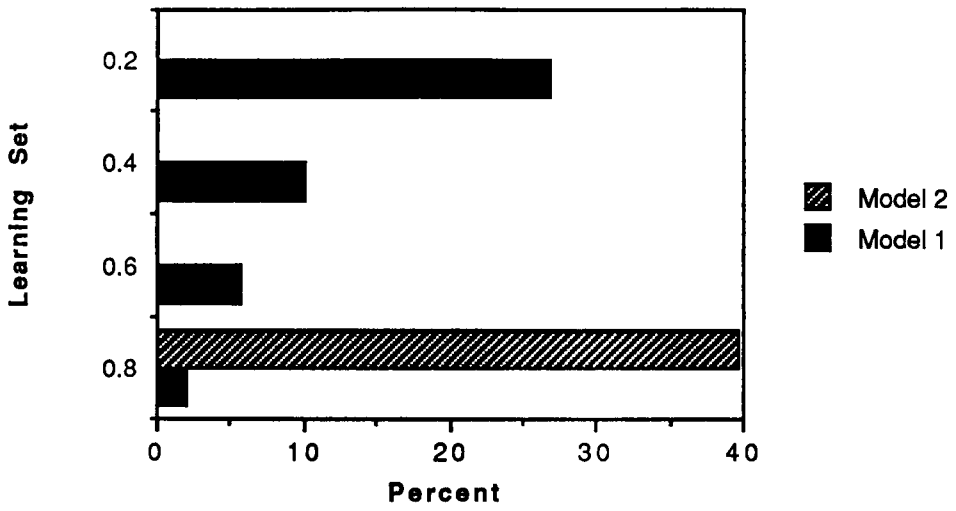


Figure 12

Percentage of Incorrect Translated Words for Models 1 and 2

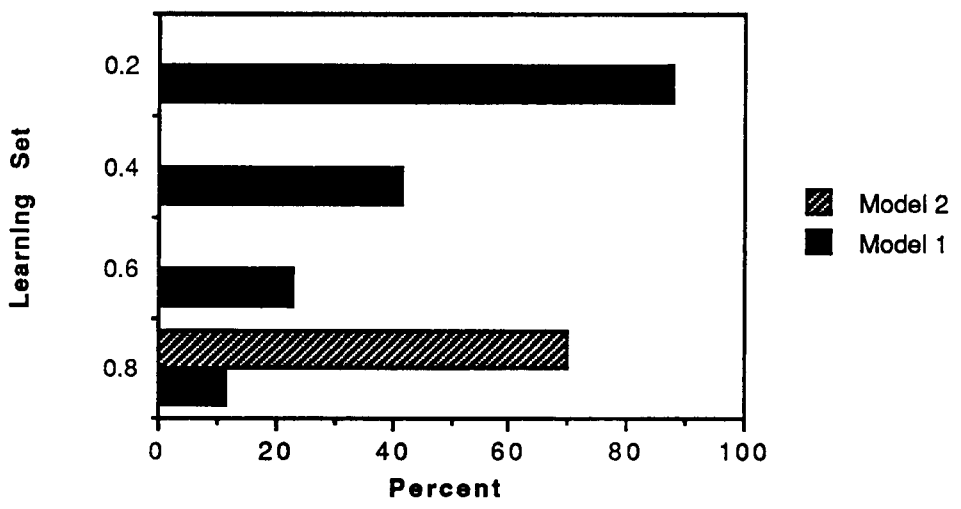


Figure 13

Percentage of Incorrect Translated Sentences for Models 1 and 2

3.5 Sequential Recurrent Network Training Methods

In a recurrent neural network, a sentence is entered one word at the time. Previous words and their order are internally represented and compressed in the memory layer. When the last input word is entered, learning occurs; weights are changed according to the difference between the target and the output values. The main problem in learning with recurrent neural networks is that only the last input word is present at the time of learning. This means that the input for all but one of the words will be 0 at learning time. With all but one of the input units having an activation of 0 during the learning process, it becomes extremely difficult to associate a particular output pattern to an input pattern.

To solve this problem, and therefore improve learning, the activation process was divided into two phases. In the first phase, the input sentence was presented one word at a time to allow an internal representation of the sentence to be encoded in the memory units. In the second phase, all the words that occurred in the sentence were presented simultaneously. When the learning process takes place, the network then contains direct information on which input units were present in that sentence, and therefore the connection weights to the input layer can be adjusted to properly produce the desired output. In short, the word order is compressed in the memory layer during the first phase, and that information along with the present sentence input words is available in the second or learning phase.

This approach worked satisfactorily for the first two experiments, feature extraction and sentence comparison. For the translation task, however, learning proved to be a very difficult process. Initially, the network was trained with only 8 input words, providing a total of 32 sentences for the learning set. Training was very slow, and even after much training, the network could not learn the correct translation of 4 sentences. The main problem was the

amount of information the network had to remember. The network had to remember not only the order of the input words and the correspondence between English and Spanish words, but it also had to produce the corresponding Spanish sentence one word at the time. The network had to "know" the order of the output words and which words had already been generated.

This problem was directly related to the choice of the μ parameter in the recurrent network. In the sequential recurrent model to translate from English to Spanish, there are two types of memory layers. The first memory layer is used to encode the input sentence as is entered one word at the time. The second memory layer is used to keep track of the output words already produced.

The two memory layers perform different functions, and they perform better with different values of the μ parameter. The μ parameter determines the effect of past activations on the memory units. A value of μ equal to 0 was preferred for the first memory layer while a value of μ equal 0.6 was preferred for the second memory layer. These values of μ were determined through trial and error. Since the activation of a memory unit is set equal to the copy of the activation from the hidden layer plus μ times the unit's previous activation, μ equal to 0.6 means that the previous activation of the memory units are taken into account when computing the state of the memory layer. This allows the network to keep track of the history of the words already output. To be able to use two different values for μ, the training process was divided into two steps (see figures 14 and 15).

In the first step, the network learned to encode the input sentences as they were entered one word at a time. The parameter μ was assigned a value of 0, and the network was trained until it reached a local minimum. In the second step, a new network model was used. In this network model, the input sentence was presented already encoded. A sentence was encoded

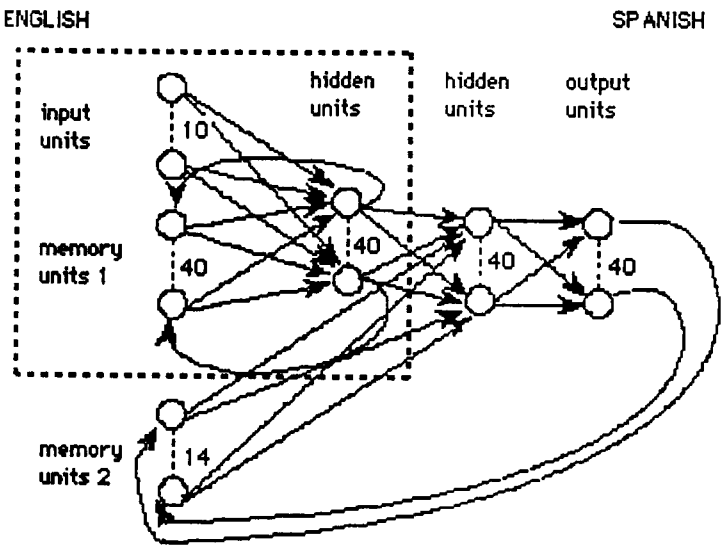


Figure 14

Step 1: Learning Using a Parameter μ Equal to 0

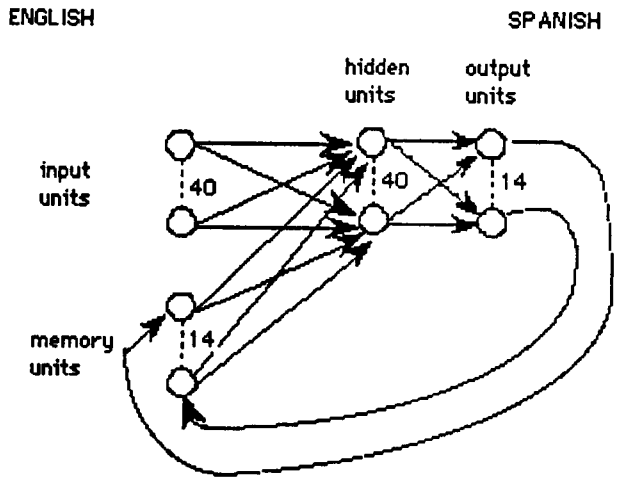


Figure 15

Step 2: Learning Using a Parameter μ Equal to 0.6

as a state of activations of the first hidden layer. The network obtained in step one was used to calculate those states. A parameter μ equal to 0.6 was used during the learning process to improve the performance of the second memory layer.

This approach improved the learning process considerably, but it was a very time consuming process, since the network was very large and two different training sessions were required.

CHAPTER 4

EXPERIMENTAL RESULTS AND CONCLUSIONS

In this section, some conclusions are derived from the experiments described earlier involving feed-forward and sequential recurrent neural networks. This is followed by a brief discussion on the possibility of scaling up such models.

4.1 Overall Performance

Both feed-forward and sequential recurrent network models showed a good ability to deal with language processing tasks such as feature extraction, sentence comparison, and translation between English and Spanish. Some of the conclusions arrived during these experiments are presented:

Momentum and learning rate parameters

A momentum of 0.5 and a learning rate of 0.2 was primarily used on the training of feed-forward models. Although in some cases they can lead to a local minimum, they allow a much faster convergence in the learning process. When oscillation occurred, a momentum of 0.3 and a learning rate of 0.1 were used.

Parameter μ in sequential recurrent networks

A parameter μ equal to 0 was used for recurrent networks whose memory units received feed-back connections from a hidden layer. A parameter μ equal to 0.6 was preferred when the feed-back connections came from the output units.

Training

The sequential recurrent network did not learn perfectly in the translation task and some instances of the sentence comparison. The size of the input patterns and a smaller learning rate caused the recurrent network to learn very slowly.

Sequential output

The results from these experiments indicate that sequential recurrent networks do not perform very well generating sequential output, such as in the case of the translation between English and Spanish, where the translated sentence had to be produced one word at a time. On the other hand, when the output was to be produced in one step, the recurrent networks proved to be similar or even better than the feed-forward networks.

Number of connections

The sequential recurrent models required several times fewer connections than the feed-forward models except for the translation task. Sequential recurrent networks offer an alternative for a more compact encoding of sentences resulting in a smaller network. When the difference in the number of connections in the recurrent and feed-forward models is large enough, the training process is shorter using the recurrent model. Such was the case in the sentence comparison task, where the recurrent model was approximately ten times smaller than the feed-forward model.

4.2 Generalization

Overall, the results indicate a good ability for both networks to generalize when presented 40% or more of the sentences in the learning process.

In the translation task, the sequential recurrent network generalized poorly after being taught with 80% of the sentences. This occurred partially because the network had not only to encode the incoming English sentence, but also produce the output Spanish sentence one word at a time; the network had to learn the correct order and somehow keep track of which word was to be output next.

Except for the case of translation between English and Spanish, the sequential recurrent models were comparable if not better than the feed-forward models. Sequential recurrent networks seem to offer a good alternative to deal with language processing tasks, not only for their smaller size, but also for their good ability to generalize on novel inputs.

4.3 Scaling

One important issue in the application of neural networks is the ability to scale up these models so that they can handle larger input sets. It is difficult to predict what kind of network could handle a specific input/output mapping, mainly because the design of such networks is experimental -- essentially trial and error. This task becomes even more difficult due to the different possibilities in designing a model and the different choices of parameters during the learning process.

In natural language processing, the feed-forward models used in previous experiments require a large number of input units to encode a sentence. Encoding of a sentence is provided by word slots, each word slot containing n units, where n is the total number of input words. This puts a limitation not only on the number of input units required to represent a sentence, but also on the maximum length of the sentence. Recurrent networks, on the other hand, provide

better expectations in scaling up such models. An increase in the number of input words would result in the same increase in the number of input units. More memory units would be needed to encode a larger number of sentences, but this increment in the number of memory units should not be very large, since sentences are encoded through the activations in the memory layer; as the memory layer increases, the number of possible combinations of states in the memory layer grows exponentially, being able in theory to assign each combination of states to a different input sentence.

A few set of experiments were conducted to get a feeling for the scaling up of the recurrent model for the feature extraction task. The results suggest that to encode the 116 sentences from 11 different input words, a memory layer of at least 10 units was needed. However, this number does not increase dramatically as more input words are added. With 12, 13 and 14 input words producing 155, 195 and 215 sentences, respectively, a total of 15 memory units suffice to extract the correct features from the input sentences.

CHAPTER 5

FUTURE WORK

There is a need for a better understanding of neural networks. Because there are no guidelines for the design and choice of parameters in a neural network, neural network models are currently developed by trial and error, that is, by designing different models and testing them experimentally. More work on mathematical foundations and statistical methods would bring us closer to understanding the possibilities and limitations of neural network models. In addition, new and better methods for training recurrent neural networks, along with the appropriate hardware, would speed up the research process immensely.

A combination of neural network learning and some "pre-wired" knowledge would allow the network to concentrate in solving more complicated problems without having to learn everything by experience. This is particularly true in natural language processing where some of the knowledge from computational linguistics could be encoded into the neural network to make the learning of a language processing task easier.

Neural networks could be applied to perform a task within a language processing entity such as a parser or a program to translate between two natural languages. Such an application would permit much faster processing, since neural networks are inherently very fast, given adequate hardware. All computing is done in parallel, and the operations performed are very simple.

Certain neural networks when applied to natural language processing exhibit behavior similar

to humans. A better understanding of the processing of natural language by humans might be achieved through the study of such networks.

NOTES

1. Scientific Symposium on Syntax and Semantics for Text Processing and Man-Machine Communication, *Natural Language at the Computer*, Springer-Verlag, New York, Berlin, Heidelberg, p. 1 (1988).

2. McCulloch, Warren and Pitts, Walter, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics 5, pp. 115-133 (1943).

3. Rumelhart, D.E. and McClelland, J. L., *On Learning the Past Tenses of English Verbs,* Parallel Distributed Processing: Exploration in the Microstructure of Cognition, vol. 1: Foundations, p. 267 (1986).

4. McClelland, J.L., *Parallel Distributed Processing and Role Assignment Constraints,* TINLAP-3, Theoretical Issues in Natural Language Processing 3, p. 78 (1987).

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Ahuja, S. B., Soh, W. and Schwartz, A., *A Connectionist Processing Metaphor for Diagnostic Reasoning*, International Journal of Intelligent Systems, vol. 4, pp. 155-180, 1989.

[2] Allen, R. B., *Several Studies on Natural Language and Back-Propagation*, Proceedings of the IEEE First Annual International Conference on Neural Networks, vol. II., 1987.

[3] Bruck, J. and Sanz, J., *A Study on Neural Networks*, International Journal of Intelligent Systems, vol. 3, pp. 59-75, 1988.

[4] Charniak, E., *Connectionism and Explanation*, TINLAP-3, Theoretical Issues in Natural Language Processing 3, New Mexico, 1987.

[5] DARPA, *DARPA Neural Network Study*, AFCEA International Press, Fairfax 1988.

[6] Golden, R., *Representating Causal Schemata in Connectionist Systems*, Proceedings of the Eighth Annual Conference of the Cognitive Science Society, Amherst, Massachusetts, 1986.

[7] Hinton, G. E., *Learning distributed Representations of Concepts*, Proceedings of the Eighth Annual Conference of the Cognitive Science Society, Amherst, Massachusetts, 1986.

[8] Hinton, G. E., *Connectionist Learning Procedures*, Artificial Intelligence, vol. 40, pp. 185-234, 1989.

[9] Johnson, M. L., *Mind, Language, Machine: Artificial Intelligence in the Poststructuralist Age*, MacMillan Press, London, 1988.

[10] Jordan, M. I., *Attractor Dynamics and Parallelism in a Connectionist Sequential Machine*, Proceedings of the Eighth Annual Conference of the Cognitive Science Society, Erlbaum, Hillsdale, New Jersey, 1986.

[11] Lehnert, W. G., *Possible Implications of Connectionism*, TINLAP-3, Theoretical Issues in Natural Language Processing 3, New Mexico, 1987.

[12] MacLennan, B. J., *Technology-Independent Design on Neurocomputers: The Universal Field Computer*, Proceedings of the IEEE First Annual International Conference on Neural Networks, vol. III., 1987.

[13] McClelland, J. L. and Kawamoto, A., *Mechanisms of Sentence Processing: Assigning Roles to Constituents*, from *Parallel Distributed Processing: Exploration in the Microstructure of Cognition, vol. 2*, MIT, Press, Cambridge, Massachusetts, 1986.

[14] McClelland, J. L. and Rumelhart, D. E., *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*, The MIT Press, Cambridge, Massachusetts and London, England, 1988.

[15] McClelland, J. L., *Parallel Distributed Processing and Role Assignment Constraints*, TINLAP-3, Theoretical Issues in Natural Language Processing 3, New Mexico, 1987.

[16] Miikkulainen, R. and Dyer, M. G., *A Modular Neural Network Architecture for Sequential Paraphrasing of Script-based Stories*, CSD-890010, January 1989.

[17] Nolfi, S. and Parisi, D., *Learning to Understand Sentences in a Connectionist Network*, Proceedings of the IEEE First Annual International Conference on Neural Networks, vol. II., 1987.

[18] Potter, C. W., *A Study of Factors Affecting the Ability of a Back Propagation Net To Recognize Alphabetic Characters and Produce Specific Symbols*, Master's Thesis, University of Tennessee, Knoxville, December 1988.

[19] Rumelhart, D. E. and McClelland, J. L. *On Learning the Past Tenses of English Verbs*, from *Parallel Distributed Processing: Exploration in the Microstructure of Cognition, vol. 1: Foundations*, MIT, Press, Cambridge, Massachusetts, 1986.

[20] Schank, R. and Kass, A., *Natural Language Processing: What's Really Involved?*, TINLAP-3, Theoretical Issues in Natural Language Processing 3, New Mexico, 1987.

[21] Scientific Symposium on Syntax and Semantics for Text Processing and Man-Machine Communication, *Natural Language at the Computer*, Springer-Verlag, New York, Berlin, Heidelberg, 1988.

[22] Sejnowski,T. J. and Rosenberg, C.R., *Parallel networks that learn to pronounce English text*, Complex Systems, vol. 2, pp. 367-372, 1987.

[23] Tank, David W. and Hopfield, John J. *Collective Computation in Neuronlike Circuits*, Trends in Computing, Scientific American, vol. 1, 1988.

[24] Touretzky, D. S., *Connectionism and Compositional Semantics*, CMU-CS-89-147, 1989.

[25] Waltz, D. L., *Connectionist Models: Not Just A Notational Variant, Not A Panacea*, TINLAP-3, Theoretical Issues in Natural Language Processing 3, New Mexico, 1987.

APPENDICES

APPENDIX A

EXAMPLES OF TRAINING SETS

Feature extraction:

For the syntactic feature extraction task, 269 sentences were generated using the following 15 words:

mary, john, speaks, to, is, spoken, by, cat, sees, seen, the, dog, listens, understands, and *bobby.*

For each sentence, there are a total of 6 syntactic features:

A) whether the sentence was in active or passive voice,

B) whether the subject contained an article (e.g. 'the') or not,

C) whether the object contained an article (e.g. 'the') or not,

D) whether the order of the sentence was normal or not,

E) whether the sentence was of the type Subject-Verb (no object) or not, and

F) whether the sentence contained a preposition (e.g. 'to') or not.

Here are some examples of the training set:

Sentence	A	B	C	D	E	F
john sees the cat	1	0	1	1	0	0
to the dog speaks bobby	1	0	1	0	0	1
the cat is spoken to by john	0	0	1	1	0	1
the cat listens	1	1	0	0	1	0
the cat understands the dog	0	1	1	1	0	0

Sentence comparison:

For the sentence comparison task, 400 pairs of sentences were generated using the following 10 words:

john, speaks, to, is, spoken, by, cat, sees, seen and *the.*

A value of 1 was used to indicate that the two sentences meant the same, a value of 0 was used otherwise. Here are some examples of the training set:

Pairs of sentences	Same meaning
the cat sees john john sees the cat	0
the cat is spoken to by john john speaks to the cat	1
to the cat speaks john john speaks to the cat	1
the cat is spoken to by john to the cat speaks john	1

Translation between English and Spanish:

For the translation task, 9 English words were used to produce 162 sentences. A total of 13 Spanish words were needed to translate those sentences. The words used were the following:

English: *joseph, cat, cow, the white, sees, speaks, to* and *a.*

Spanish: *jose, gato, vaca, la el, blanco, blanca, ve, habla, a, un* and *una.*

Here are some examples from the training set:

English	Spanish
joseph speaks to the cow	jose habla a la vaca
joseph speaks to the cat	jose habla al gato
joseph sees the cat	jose ve el gato
joseph sees the white cat	jose ve el gato blanco
joseph sees joseph	jose ve a jose
the white cat sees the cow	el gato blanco ve la vaca
the white cow sees joseph	la vaca blanca ve a jose

APPENDIX B

NETWORK DEFINITION FILES

The neural networks used in this work were implemented using the software provided in "Explorations in Parallel Distributed Processing" by J. L. McClelland and David E. Rumelhart, in particular the back propagation (bp) program.

In the definition of sequential recurrent networks, the memory units are not differentiated from the input units. The feed-back connections are specified as part of the input pattern; a negative input to a memory unit, say *-n*, indicates that that memory unit receives its input from unit number *n*. In order to avoid learning while a sentence is being entered one word at the time, the target values are set to -1 until the last word in the sentence is entered, in which case the target values are presented. For more detail, see pp. 156 - 158 in "Explorations in Parallel Distributed Processing" by J. L. McClelland and David E. Rumelhart.

The following network definition files were used in this work:

Feature extraction:

```
definitions:
nunits 146
ninputs 120
noutputs 6
end
network:
%r 120 20 0 120
%r 140 6 120 20
end
biases:
%r 120 26
end
```

Model 1: Feed-forward

```
definitions:
nunits 61
ninputs 35
noutputs 6
end
network:
%r 35 20 0 35
%r 55 6 35 20
end
biases:
%r 55 6
end
```

Model 2: Recurrent

Sentence comparison:

```
definitions:
nunits 198
ninputs 160
noutputs 1
end
network:
%r 160 30 0 160
%r 190 7 160 30
%r 197 1 190 7
end
biases:
%r 160 38
end
```

Model 1: Feed-forward

```
definitions:
nunits 66
ninputs 40
noutputs 1
end
network:
%r 40 10 0 20
%r 50 10 20 20
%r 60 5 40 20
%r 65 1 60 5
end
biases:
%r 40 26
end
```

Model 2: Recurrent

Translation between English and Spanish:

```
definitions:
nunits 209
ninputs 72
noutputs 117
end
network:
%r 72 20 0 72
%r 92 117 72 20
end
biases:
%r 72 137
end
```

Model 1: Feed-forward

```
definitions:
nunits 159
ninputs 64
noutputs 14
end
network:
%r 64 40 0 50
%r 104 40 50 54
%r 144 14 104 40
end
biases:
%r 104 54
end
```

Model 2: Recurrent

VITA

Fermin Soriano Salkjelsvik was born in Malaga, Spain on July 1, 1963. He attended elementary school at Nuestra Señora del Buen Consejo, Madrid, and graduated from INB Aluche High School in June 1982. The following September he entered the University of Bergen, Norway, and in June 1986 he received a Bachelor of Science degree in Computer Science. Between September 1985 and March 1987, he worked for The University of Bergen, first as a computer operator, and later as a computer programmer. In September 1987, he entered the Master of Science degree program in Computer Science at the University of Tennessee, Knoxville. He married Elizabeth McFarlin Jones in June of 1988. As a graduate teaching assistant he taught FORTRAN and Computer Organization. The author was member of the soccer teams of both The University of Bergen, Norway, and the University of Tennessee, Knoxville.