INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Université d'Ottawa · University of Ottawa

THE APPLICATION OF NEURAL NETWORKS TO NATURAL LANGUAGE TRANSLATION

by

© Bruce McHaffie

Advisors:

Brian Harris
School of Translation and Interpretation

and

Mario Marchand, Ph.D.
School of Information Technology and Engineering

Dissertation submitted to the School of Graduate Studies and Research of the University of Ottawa in partial fulfilment of the requirements for the degree of Master of Arts in Translation

University of Ottawa
Faculty of Arts
School of Translation and Interpretation
1997

ABSTRACT

The characteristics of automated learning and generalization, and of graceful degradation in the face of unforeseen input, give neural networks interesting potential for machine translation (MT). However, the field of connectionist MT has been little explored by researchers. This thesis provides an introduction to neural network concepts and summarizes and reviews the research in the field of connectionist MT. It describes the building, training and testing of TransNet, an embryonic neural network that translates weather reports from English into French. TransNet uses an innovative *bigram* (word pair) data representation which makes it possible to take some account of word order in the processing.

RÉSUMÉ

De par leurs capacités d'apprentissage et de généralisation automatiques, et de détérioration graduelle face aux données de nature imprévue, les réseaux de neurones présentent un potentiel fort intéressant pour l'avenir de la traduction automatique. Pourtant, ce potentiel a été très peu exploré par les chercheurs dans le domaine. La présente thèse commence par une introduction aux concepts fondamentaux des réseaux de neurones et une revue de la recherche menée jusqu'à ce jour. Elle décrit ensuite la création, l'entraînement et la mise à l'essai d'un réseau de neurones rudimentaire, TransNet, spécialement conçu pour traduire des bulletins météorologiques de l'anglais au français. Ce dernier exploite une façon innovatrice de représenter les données textuelles en découpant les textes en *bigrammes* (couples de mots), ce qui permet de conserver tout au long du traitement une trace de l'ordre des mots dans le texte.

ACKNOWLEDGEMENTS

Heartfelt thanks to my thesis advisors Prof. Brian Harris and Dr. Mario Marchand for their insightful comments and suggestions. Their openness to inter-disciplinary research demonstrates a strong commitment to academic ideals. I also thank my examiners, Dr. Ingrid Meyer and Dr. Stanislaw Szpakowicz, most of whose criticisms have been taken account of in my final text.

TABLE OF CONTENTS

INTRODUCTION

The purpose of this thesis was to investigate the applicability of neural networks to natural language translation. There were three aspects to our investigation: exploring the technology that underlies neural networks to find out if it could be applied to translation; reviewing the previous research in the field; and designing, training, and testing a translating neural network ourselves. The topics are presented in that order, although in some cases they span chapter boundaries.

Chapter 1: History of Neurocomputing. This chapter presents a brief history of the topic.

Chapter 2: Description of Neural Network Operation. This chapter uses examples to describe how neural networks work. It also describes the different techniques that can be used to train a neural network, and explores the different network architectures that can be used to perform different types of tasks.

Chapter 3: Properties of Neural Networks. This chapter discusses the characteristics that set neural networks apart from other types of computational approaches; they include learning, generalization, and parallelism. The concepts are explained in the context of their potential usefulness in computerized systems that translate natural language. The chapter also describes some of the weaknesses of the connectionist approach and deals with the question of whether artificial neural networks should be likened to natural ones such as those in the human brain. This was felt to be relevant because many non-technical people still consider that the artificial networks simulate the natural ones.

Chapter 4: Connectionist Natural Language Processing. This chapter outlines the main differences between the connectionist approach and the rule-based approach in natural language translation. It also introduces the challenges inherent in representing natural language data to a neural network.

Chapter 5: Literature Review of Connectionist MT. This chapter reviews the literature in the field of connectionist natural language translation. Research on connectionist natural language processing that was felt to have a bearing on translation is also included here. The chapter also brings to light some of the oversights of previous research in this little-explored field.

Chapter 6: Experiment. This chapter describes the neural network TransNet that was built to test the applicability of neural networks to natural language translation. TransNet was constructed using the Xerion neural network simulation environment (designed and programmed by researchers at the University of Toronto). TransNet was designed to translate a small body of weather reports from English into French. This section also discusses the TransNet results.

Chapter 7: Recommendations. This chapter presents our recommendations for further research.

Chapter 8: Conclusion. This chapter presents our views on the applicability of neural networks to natural language translation, supported by our investigation into the promises of connectionism, our review of current literature, and our own experiment.

CHAPTER 1: THE HISTORY OF NEUROCOMPUTING

The history of neurocomputing dates back 54 years to 1943. The principle that simple neural networks could compute arithmetic or logical functions was first set out in a paper by Warren McCulloch, a neurophysiologist, and Walter Pitts, a mathematician, in 1943 (Hecht-Nielsen 1990, 14). Other researchers such as the mathematician John von Neumann also saw promise in research on computers inspired by the operation of the brain, and von Neumann himself used the McCulloch-Pitts paper in a course on the theory of computing machines (Nelson and Illingworth 1990, 26). In 1949, psychologist Donald Hebb proposed a specific learning mechanism for the synapses of biological neurons in an attempt to account for classic psychological conditioning in animals. Spurred by such insights in biology and psychology, Frank Rosenblatt, a psychologist turned connectionist researcher, constructed the first neurocomputer (the Mark I Perceptron) in 1958. The Perceptron, so named because it was thought by its inventor to model the biological mechanisms involved in human perception (Muller and Reinhardt 1990, 14), was built as a pattern recognizer—it classified characters that were presented to it through an array of photoconductors. The general perceptron model has also been successfully applied to other problems such as image classification and robot vision systems (Nelson and Illingworth 1990, 115). The very first neural network applied to a real-world problem was MADALINE (Multiple ADAptive LINear Elements), developed by Bernard Widrow and Marcian Hoff in 1959 (Nelson and Illingworth 1990, 28). It was used to eliminate echoes on phone lines. For the most part, though, what these machines did was much less interesting than how they did it.

In essence, and unlike any automaton before them, neural networks developed their own ways of solving problems, being supplied only with the input and desired output of a system. While actual

applications were modest, the self-organizing nature of the networks was believed to mimic the process of human learning, and public reaction was predictably overwrought. Rumours flew concerning the imminent arrival of thinking machines, and human-like automatons: one Oklahoma newspaper announced "Frankenstein Monster Designed by Navy Robot That Thinks" (Nelson and Illingworth 1990, 29.) This recalls the extraordinary optimism of some early machine translation researchers, though expectations were never realized.[1] By 1969, the serious learning limitations of Rosenblatt's perceptron model had been summarized in a book called "Perceptrons" by Marvin Minsky and Seymour Papert. They showed that Rosenblatt's network, which contained only one layer of input units, was unable to solve several very basic problems in information processing. One example was the *exclusive-or* (XOR) logical function which required the output unit of a network to be activated if, and only if, one of two input units is activated. Rosenblatt's networks were incapable of modelling this and other functions easily modeled by conventional circuits. Interestingly, Minsky and Papert also noted that a neural network containing a layer of units between the input and output layers would allow such problems to be overcome, but at the time there existed no theoretically sound algorithm by which multi-layered networks could be trained to produce desired results (Carling, 1992, 106). While the publication of this work is often cited as the main cause of the subsequent demise of neural network research, in fact Rosenblatt was already aware of the problems underscored in the Minsky-Papert book, and was actively working to resolve them. However, no solutions were forthcoming, and this lack of new ideas in the neurocomputing community combined with Minsky and Papert's influence brought work on neural networks to a virtual halt.

---

1. "The [Georgetown] demonstration of Russian-English MT [on 7 January 1954] received wide publicity; Sheridan and Dostert [the architects of the system] lectured widely on the system and the potential future of MT" (Hutchins, 1986, p. 37).

Between 1969 and 1982, little direct research was undertaken in the field, although work did proceed under related headings such as adaptive signal processing and biological modelling (Hecht-Nielson 1990, 19). By the early 1980s, however, neurocomputing was enjoying a resurgence, in part because of sponsorship by the U.S. Government's Defense Advanced Research Projects Agency (DARPA). Also instrumental was the work of John Hopfield, a well-respected physicist whose papers and lectures on neural networks attracted interest from around the world (Nelson and Illingworth 1990, 30). Crucial to the advancement of the field was the (re)-discovery of the backpropagation training algorithm—an efficient way of training multi-layered networks. This development overcame the major objections raised by Minsky and Papert in 1969, and neural networks were now capable of exploring solutions to a large set of problems previously closed to them. The backpropagation training technique was described in a 1986 paper by Hinton and Williams (Rumelhart and McClelland 1986). These two researchers are responsible for the current widespread use of the technique, although the principles had been set out earlier by P.J. Werbos in a Master's thesis in 1974 (Hecht-Nielson 1990, 125). In "Neural Computing", Phillip Wasserman identifies such duplication of effort as a significant problem in neurocomputing due to the "interdisciplinary nature of the subject" (Wasserman 1989, 44), He rightly goes on to point out that "neural network research is published in books and journals from such diverse fields that even the most diligent researcher is hard pressed to remain aware of all the work". The experience of the present author has corroborated Wasserman's finding. Another important event occurred in 1986 with the publication of what was to become the connectionist bible—Rumelhart and McClelland's "Parallel Distributed Processing". This seminal work provided a common and easily accessible pool of information for other researchers and helped further accelerate the pace of research in

connectionism. In 1987 came the first open conference on the topic, the First IEEE International Conference on Neural Networks.

CHAPTER 2: DESCRIPTION OF NEURAL NETWORK OPERATION

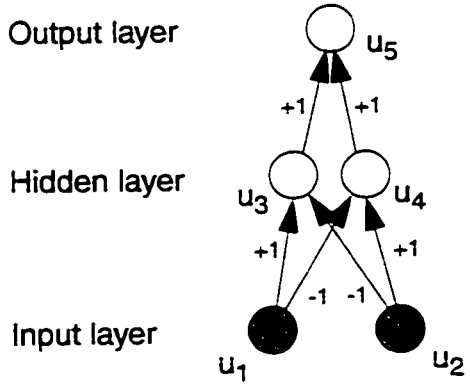
To understand what makes neural networks worthwhile, we should first understand how they work. Briefly, a neural network is a processing structure consisting of a set of processing elements called *units* or *nodes* and a set of unidirectional connections[2] between the units (Gallant 1993, 3). The network is a set of these units organized into layers. With the exception of the input layer, each layer receives as input the output signals from units in previous layers. The input layer receives its input from outside the network. The receptor units in each layer then compute the weighted sum of the incoming signals and produce their own output signal (called an activation) to serve either as input to yet another layer of units, or as an output from the network. Each connection between units carries a *weight*—a positive or negative value by which the activation it transmits is multiplied. The larger the weight of a connection (positive or negative) the greater the influence a given activation will have on the receiving unit or units. A negative weight means that the link will inhibit activation in the receiving unit, while a positive weight will reinforce activation (Gallant 1993, 11). After summing its weighted input, a unit then applies an activation function to determine its output. This function serves to keep the value of the output bounded, usually within the range of 0 to 1. Things become more complex—a unit can be one of three types: input, hidden or output. As we have mentioned, input units receive their activations from outside the network. These activations represent the information to be processed by the network, and are often passed to the input units from a stored file. The input units do no summation; they simply broadcast the input data through weighted connections to the units that are next in line. Hidden units are neither input units or output units, but reside in layers between the input and output layer.

---

2. Hence neural network systems are also called *connectionist*.

They receive input from other units and provide their processed activations to yet other units. Output units receive the activations of hidden units or input units, and process them to provide the output activations for the network overall. A simple network can be seen schematically in Figure 1.

Figure 1 The XOR network



The flow of activations follows the arrows on the connections between units, starting at the input layer (u_1 and u_2), proceeding through the intermediate or hidden layer (u_3 and u_4), and on to the single unit in the output layer (u_5). It should be noted that it is customary not to include the input layer in the layer count, so that the network seen in Figure 1 would be considered a 2-layer network. In the case shown here, the data provided to the input layer (units shown as darkened circles) comes in the form of ones and zeros. Each input unit then transmits its activation across all of its outbound connections. In this case, given the input $u_1 = 1$ and $u_2 = 0$, unit u_1 provides a *1* to both its connections, while u_2 provides a *0*. These activations are then multiplied by the weights attached to each connection (the weight for each connection is shown in the Figure), resulting in

u_3 receiving a 1 * 1 = 1 from u_1 and a 0 * -1 = 0 from u_2; while u_4 receives a 1 * -1 = -1 from u_1 and a 0 * 1 = 0 from u_2. It is important to note that in our example the units in the hidden layer propagate their activations to the output layer only when their value is greater than 0. For our example then, u_5 receives input only from u_3 (1 * 1 = 1); since the activation of u_4 is -1, it contributes nothing to the activation level of u_5. Then the single output unit u_5 has the simple job of summing 1 and 0 to arrive at the value 1. This activation serves as the output for the network as a whole. This simple sequence of steps provides a solution to the exclusive-or (XOR) problem, a basic Boolean function. For this function, an input of {1,0} or {0,1} produces an output of {1}, while inputs {0,0} or {1,1} produce {0}.

2.1 Network Learning

The varied abilities of artificial neural networks stem from this simple propagation of values through a complex series of basic processing units. The configuration of network units and connections, i.e., what is connected to what, and their weights determine how a network will react to a given input. In this way these parameters serve as an abstract instruction set for solving computational problems. Of course the networks would be much less interesting if one were forced to somehow determine all the appropriate connection weights before presenting the network with data. Fortunately, mathematical algorithms have been developed that allow networks to automatically adjust their own weights as a function of the desired output of the network. Given a problem solvable by a particular network, these weights are modified iteratively as input is presented to the network. The result is a successively closer approximation of the desired output. This remarkable ability, often termed *learning*, is one of the most fascinating and powerful characteristics of neural networks.

Learning takes place in *training sessions*, during which the network is free to modify its connection weights according to a predetermined learning algorithm. The connection weights are first initialized to small random values. The learning process proceeds differently with different networks but in one common type, known as *backpropagation*, input values are presented to the input units and the desired output values presented to the output units. The learning algorithm then modifies the weights for each unit in the network so that the actual output unit activations are increasingly closer to the desired values. In an appropriately designed network, these weights will finally converge at values so that each group of inputs produces a group of outputs close in value to what is desired. These groups are known as vectors: in the example of the XOR network, {1,0} would be a possible input vector, and the target vector would always be a single value, in this case {1}. Once a network has been trained, the weights are usually frozen so that no more learning takes place. At this point the network is ready to process previously unseen input and it no longer attempts to update its own weights. This is the testing stage, and for some applications it is the time to evaluate the network's ability to correctly process data not submitted to it during training.

2.2 Learning Types

Networks can be trained using many different algorithms, but there are basically two types of learning that can take place: supervised and unsupervised. The first type is supervised learning, in which the network is always presented with both input and target vectors. Taken together, these vectors are known as training pairs. When presented with a training vector, the network compares the value of the actual output vector to that of the target vector. It then changes the weights on its links according to an algorithm that seeks to minimize the difference between the actual and target output vectors. The training pairs are applied one after another, and the connection weights modi-

fied until the network output has reached an acceptable level of accuracy with respect to the target vectors. This is the type of training that will be used in the translation networks investigated in this thesis.

For the sake of completeness, we will mention here the other major type of training, which is unsupervised. Unsupervised training uses no target vector for the output of the network. The training algorithm is designed to adjust connection weights so that the output vectors of similar input vectors are themselves similar; the network's output is thus guided by the structure of the input data. As a result, the network automatically extracts significant properties of the input vectors which can then be used as a basis for classification in applications such as image processing (Wasserman 1989, 23).

2.3 Learning Algorithms

As mentioned, there is a wide variety of learning algorithms, the simplest being perhaps the Hebbian learning algorithm. This is a supervised learning algorithm based on a mathematical statement of the principle that a connection between two units be strengthened (i.e., the connection weight raised) whenever both units are simultaneously excited (Rumelhardt and McClelland 1986, 36). An exhaustive examination of these algorithms is beyond the scope of this work. For present purposes, only the backpropagation learning algorithm will be discussed. While a complete description of backpropagation learning requires knowledge of differential equations, the principles upon which it rests are relatively simple.

Under backpropagation learning, network weights are initially assigned small random values. A training pair is then selected, and the activations of all units including the output units are com-

puted by the network. Next, the magnitude of weight change necessary to obtain a small decrease in overall network error is calculated for each unit by the backpropagation module associated with the network. Finally, the connection weight for each unit is automatically adjusted in proportion to this calculated value. These steps are then repeated for every pair of vectors in the training set until the overall network error for the training set becomes acceptably small. The overall network error for the training set can be calculated in several different ways, but it is always directly related to the difference between the target and actual output values for each training pair in the training set. At this point, the algorithm is said to have converged, and the output of the network will correspond closely to the target output.

2.4 Network Architecture

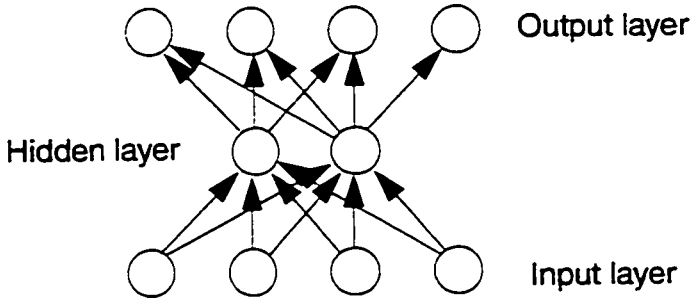
The connection weights in a network are responsible in part for the network's behaviour, but they are not the sole determiners. Another important element is the architecture of the network—what is connected to what, and how. While an infinite number of architectures is possible in principle, we will briefly examine two general types: feedforward and recurrent. These two network types, as we shall see, have application in the field of connectionist natural language processing, and in our specific case, translation.

2.4.1 Feedforward and recurrent networks

The multilayered feedforward network is a network with at least one hidden layer, in which all connections between units lead towards the output layer. If every unit in each layer is connected to every unit in the next layer, the network is said to be *fully connected*.

Figure 2 shows a 2-layer fully connected feedforward network.

Figure 2 Fully connected feedforward network



According to the numbering convention mentioned above, the input layer does not figure in the layer count. Typically, these networks take real-value inputs and are trained using backpropagation; in fact backpropagation is so commonly used to train these networks that they are often called *backpropagation networks*. As mentioned in Chapter 1, multilayered networks have superior computational power to single-layered ones (Wasserman 1989, 18). In fact, the feedforward network with backpropagation has proven adept at a surprising variety of learning tasks. For instance, the configuration has been used navigate a vehicle down a winding road, in a network called ALVINN. The inputs in this case came from a video image of the road and a laser range finder (Carling 1992, 297). ALVINN uses 29 hidden units and 45 output units to accomplish its task. Another example of this approach is NETtalk, a feedforward backpropagation network that was trained to pronounce written English words (Gallant 1993, 230). Note that an infinite variety of feedforward networks can be created by varying the number of hidden layers and the number of units in each layer. While the number of input and output units is often dictated by the problem to be solved (for instance recognizing the twenty-six lower-case letters would likely demand at least 26 output units), the number of hidden layers and units is an experimental variable. Though

proven methods exist to train connection weights once a network has been configured, the optimal hidden layer configuration is commonly determined by trial and error (Gallant 1993, 230). Some work has been done on algorithms to automatically generate network architectures, including recent work by Marchand and Golea (see Marchand and Golea, 1993), but no broadly applicable method has yet been found (Gallant 1993, 204).

The second type of network architecture is known as *recurrent*. These networks differ from feed-forward networks in that connections do not all proceed from the input towards the output layer. Some link the outputs of units to inputs in the same or previous layers. When the output is redirected to the input layer, each successive output depends not only on the most recent input (as with nonrecurrent nets) but also on the state of the network reflected in previous outputs. This phenomenon gives the network a sort of short term memory, which can be used in many different ways. For instance, recurrent networks called Hopfield networks have been used in associative memory models (Wasserman 1989, 94), in which a partial set of data (image, word, etc.) leads to the recovery of a complete related memory. Called auto-associators, these networks are trained to reproduce their inputs at the output layer. Data sets with missing or incomplete information are then input to the network, which is able to infer the missing information. As we shall see, recurrent networks have also found application in natural language processing. J. Elman, for instance, has implemented a recurrent network which contains connections between a hidden and a *context* layer, the latter serving as a holding tank for previously generated hidden unit activations (Elman 1990a). Elman used this model to account for the sequential nature of written language. His network was designed to predict the next word in a sequence of words, effectively using context to predict word order.

CHAPTER 3: PROPERTIES OF NEURAL NETWORKS

The XOR network example given above is extraordinarily simple in that it models a Boolean function which has a limited set of possible inputs and outcomes, and operates only with discrete input and output values. Were neural networks limited to representing such functions, the field would be all but ignored. After all, a simulation of the XOR gate can be accomplished by a set of rules embodied in a few lines of programming code. Fortunately, as we have seen briefly, neural networks with much more interesting properties can be created. Instead of Boolean values, input units might represent an image, or the phonetic characteristics of a word. The output units might then serve to categorize an image, or identify its content, or pronounce a word. We will look shortly at the ways in which these types of information can be presented to the network. First though, we should examine the unique abilities inherent in the simple operation of neural networks.

3.1 Learning

Perhaps the most intriguing and powerful of their abilities is learning. In unsupervised training, networks modify their weights so that similar input patterns produce similar output patterns (i.e., input patterns become classified according to their output). During supervised learning, networks alter their weights so that input patterns are mapped onto specified output patterns. The network performs this feat without the benefit of explicit rules to govern the mapping—the rules, so to speak, are inferred from the data. So, given an appropriate network configuration, the network can automatically learn how to map input to output when presented with a sufficiently large set of examples. Neural networks become an attractive alternative to conventional artificial intelligence (AI) strategies where the solution to a problem cannot easily be codified in a set of rules. In this

way neural networks may help fill a gap in traditional computing methods, which excel at processing data according to a set of formalized pre-established instructions. Many cognitive tasks such as pattern recognition and vehicle navigation are difficult to formalize, and the use of neural networks is accordingly being investigated in these areas.

This is not to say that rule-based approaches cannot be successful in such fields. Indeed, it is widely accepted that traditional computing is superior to neural networks wherever appropriate sequences of instructions can be laid out. A case in point is the DECtalk text-to-speech conversion system which reads aloud English text. Unlike NETtalk mentioned above (page 13), DECtalk uses a list of rules drawn up by a team of linguists. In comparisons it outperformed NETtalk by a small margin (Gallant 1993, 233). However, while DECtalk was perfected over a period of 20 years, NETtalk was constructed and trained in a few months (Carling 1992, 269). This typical reduced implementation effort is another important quality of neural networks. Where cognitive tasks are concerned, Muller summarizes the difference between the two as follows: "It is only fair to state that often many man-years of analysis are required before the problem is sufficiently well-understood to be accessible to computational treatment. On the other hand, a neural network may be able to learn from a large number of examples within a few hours of synaptic strength adjustment" (Muller 1990, 18). This power alone would make it worthwhile taking a look at neural networks for machine translation (MT), given that it usually takes several years to develop satisfactory MT systems.

3.2 Generalization

Neural networks are also able to generalize from training data—that is, to produce correct output when presented with data they have not encountered previously. For instance, a network trained to

determine the gender of French nouns, and taught to classify "adresse" as feminine might also be expected to correctly classify the word "tendresse" even though it had never encountered the word before. In fact a small network trained as a preliminary step in the present project was able to make such a generalization. Hand in hand with this characteristic goes the ability of the network to process noisy input, i.e., input which contains errors. In natural language, the errors may be misspellings, improper capitalizations, etc. In image processing it may be unfocused images, overexposure, etc. Neural networks are naturally noise resistant—they do not break down if errors or unexpected data are encountered in the input. As long as a sufficient segment of the input is intact, the corrupted portion will be dominated by the unaffected inputs and the correct output generated.

3.3 Parallelism

Finally, neural networks are naturally parallel in the way they process data. Since processing can take place simultaneously at all the units in a network, the system is potentially faster than traditional serial computers.[3] This allows decisions based on large amounts of input data to be made in real time (Nelson and Illingworth 1990, 60). Another related advantage is the ability to process substantially different types of information simultaneously, and combine them elegantly in pursuit of a network goal. A good example of this can be seen in the ALVINN navigator (see page 13). The input layer accepts information in the form of the vehicle's distance from objects and the image of the road. These two significantly different types of input are processed simultaneously and combined in the network to produce output representing the direction in which to steer.

---

3. However, this computational advantage is only achieved if the neural network is implemented using parallel processing hardware. It is lost when a neural network is simulated on a serial computer.

3.4 Neural Network Weaknesses

Neural networks have weaknesses as well. For one thing, it is difficult to determine how a network has solved a given task. This is because it does not break the problem down into logical steps that can be examined. The result is that there is no a priori way to ensure that solutions will always be correct, or that the network has learned precisely what was intended. An example will illustrate the point. In a now infamous experiment, a neural network was trained by the U.S. military to differentiate between photographs that contained partially hidden tanks, and those that did not. What researchers discovered after an unsuccessful testing run was that the network had actually learned to differentiate pictures taken on sunny days from pictures taken on cloudy days. By coincidence, all pictures with tanks had been taken in the sunlight, while the tank-less pictures were taken under cloud cover. To date, the best way to evaluate the operation of a neural network is to present it with test cases. In contrast, the reasoning of a rule-based system can be traced back through its decision-making process, allowing that process to be evaluated.

Neither is there any guarantee that a given network will be trained in a finite amount of time (Wasserman 1989, 7), or that a given network configuration is even capable of learning to solve the problem put to it. Indeed, many training efforts fail after consuming a great deal of computer time. In some cases, the training algorithm is unable to find the best possible set of connection weights, and settles on a sub-optimal solution. Its ability to find the optimal solution is determined by a combination of network configuration, the relationship between input and target vectors and the initial connection weights.

Despite these drawbacks, neural networks have been applied successfully to a significant number and variety of complex tasks such as pattern recognition, image processing and data analysis and

control (Hecht-Nielsen 1990, 7). Their unique attributes make them particularly attractive for exploring solutions to problems that are difficult to break into logical steps, but for which large amounts of empirical data are available as is the case with translation.

3.5 Neural Networks and Biological Plausibility

One of the original motivations for neurocomputing was the desire to imbue machines with human intelligence. Rosenblatt himself is quoted as saying of his Perceptron: "for the first time we have a machine capable of original ideas" (Carling 1992, 65). However his enthusiasm was later tempered by the seemingly insurmountable obstacles to learning discussed by Minsky and Papert. Such claims as Rosenblatt's were made on the assumption that artificial neural networks operated on a basis very similar to that of the human brain. However, as knowledge about the brain was pieced together, it became increasingly apparent that artificial networks were hopelessly simple by comparison. Not only does the brain typically consist of over one hundred billion neurons, but there are over fifty different types of neuron, each specialized for different tasks. The number of connections is estimated at around 10^{15} with some links stretching a meter or more (Wasserman 1989, 12). By contrast, even modern artificial neural networks typically contain just a few thousand processing units or less, with perhaps several thousand connections. The stunning complexity of the brain has led modern-day researchers in neurocomputing to be wary of exaggerating the link between neuroscience and their own models. Hecht-Nielsen goes as far as to say that given the primitive level of understanding of how the brain works, neurocomputing models supposedly inspired by neuroscience "probably have no close relationship whatsoever to the brain" (Hecht-Nielsen 1990, 12). As a result, many researchers such as Gallant have adopted an engineering approach to their networks. They no longer strive for biological plausibility, but base

the evaluation of their models on how well a given network performs the task for which it was designed.

This caveat notwithstanding, neural networks are well suited to research in artificial intelligence—the broad field which aims ultimately to make computers proficient at such tasks as learning, understanding natural language and perceiving and interpreting images. Unfortunately the scope of this thesis does not allow us to look at neural network research in all these areas. Our concern is the application of neural networks to language and linguistic processes. In the next section we will discuss the brief history of neural networks in natural language processing (NLP) and examine why neural networks constitute an intriguing platform for investigation. We will also discuss why a connectionist approach might be promising in the realm of machine translation.

CHAPTER 4: CONNECTIONIST NATURAL LANGUAGE PROCESSING

Preliminary investigations into connectionist natural language processing (CNLP) occurred in the early 1980s. Some researchers saw in neural networks a new way to formulate solutions to the problems encountered in NLP.

4.1 CNLP vs. Rule-Based Approaches

Up until that time, almost all research had used symbolic, or rule-based, approaches to allow computers to analyze language, for instance the formalism of transformational grammar. In this paradigm, linguistic processes are broken down into sets of general rules and the rules converted into procedural computer code.[4] For instance, a traditional system for determining the grammaticality of English sentences might contain a grammar consisting of a set of allowable part-of-speech combinations. Actual text segments would be considered grammatical if they were consistent with one or more of the allowable combinations. Obviously, such a system is only as good as its rules; even correct constructions are automatically disallowed if they are not included in the grammar. And the fact is that natural language grammar is notoriously difficult to capture in a logical framework; as every second language learner knows, there is always an exception to the rule. Indeed, there is some question as to whether the subtleties of human language *can* be codified explicitly. For instance, it is not at all clear that something like the overall tone of a text would lend itself to this kind of treatment. Neural networks, on the other hand, do not require the elaboration of explicit rules; they would infer equivalencies of the rules from a set of training data, which in this case might be a natural language corpus. What rules are inferred depends on

4. A classic Canadian example in the history of MT is Q-Systems (Systèmes-q), a high-level programming language specially designed to enable linguists to formulate such transformational rules (Hutchins 1986, 224 ff.)

the network topology and the method of representing the linguistic data—a subject that will be explored in greater depth further on—as well as the actual nature of the data presented to the network. At the very least, the neural network approach might help alleviate the problem of incomplete rule sets, and reduce the large implementation effort often required to establish valid rules.[5]

Neural networks also tend to be less brittle in the face of unknown input than do their rule-governed counterparts. For instance, a rule-governed parser will be unable to produce an analysis of a sentence if the sentence contains a combination of parts of speech which is not accounted for in the parser's rules. A neural network, on the other hand, would degrade gracefully and make an attempt at analyzing the sentence—a *best guess* approach. In work done by Elman (Elman 1990a, 210), for example, a network was able to speculate about the grammatical class membership of an invented word ("ZOG") given only information on the context in which the word appeared.[6] Specifically, "ZOG" replaced the word "man" in a series of sentences, and the network grouped the word with the words "woman", "girl", and "boy", a category we might characterize as animate and human. This type of adaptability in the face of unforeseen input is inherent in a neural network system, whereas it must be added on to traditional systems. Admittedly it may not always be desirable to guess at the output; by refusing to analyze a sentence, a system based on a symbolic approach is signalling that a problem has been encountered. This event can be used to prompt a human operator to provide the right solution.[7] In general, however, a network that made good guesses would be preferable to a program that stopped short of producing any translation.

---

5. Nevertheless, and as already mentioned, testing what a neural network has learned is problematical. One advantage of conventional rules is that they can be contemplated in the abstract or with very little data..
6. This particular result can also be achieved using rule-governed approaches.
7. The METEO machine translation system exploits this very phenomenon in order to recognize sentences it cannot process and shunt them to a human translator (Chandioux 1976).

Finally, the parallel nature of neural networks makes them ideal for processing various forms of linguistic information simultaneously. Interpreting a sentence may require the integration of syntactic, grammatical and semantic information. As we saw with the ALVINN example, neural networks are eminently capable of handling the integration of different types of constraints. Processing proceeds not sequentially, on one aspect then the next, but rather on all aspects at the same time. As a result, no algorithm is required by which the three constraints must be satisfied in some order—rather, grammar, syntax and semantics might combine due to the intrinsic property of the network and, say, recommend one interpretation over another. The very fact that multiple constraints can be satisfied simultaneously gives connectionism an intuitive appeal. The difficulty for connectionist approaches lies in finding ways of meaningfully representing these different types of information to the network and of determining an appropriate network architecture.

As we have seen, then, the ability of neural networks to infer rules from data, generalize to cover unknown input, and integrate diverse types of data makes them suitable for investigating and simulating the processes of natural language. No approach to NLP is without problems, however, and neural networks are no exception. Best guesses may be unwanted, especially if they are often wrong, and finding the appropriate network architecture and data representation may be difficult. Connectionist models to this point have not been better able than other techniques to provide anything like a complete framework for processing natural language (Lingaard, Myers and Nightingale 1992, 197). However some successes have been registered in this emerging field in areas as diverse as prepositional attachment, anaphora, and especially interesting for us, natural language translation. Before we arrive at our main topic, however, we must examine the question of data

representation mentioned several times already. It is an issue central to CNLP implementations, and merits some discussion.

4.2 Representation in CNLP

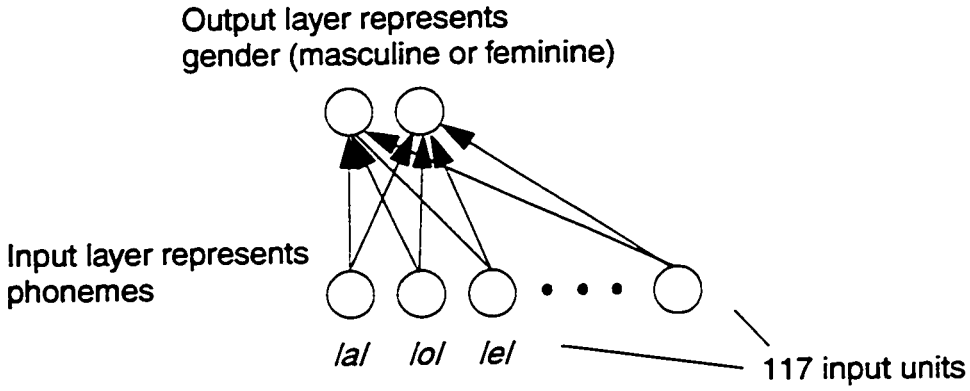
The inputs to the XOR network described in Chapter 2 represent Boolean arguments, which may only be assigned two values—*true* or *false*. These two states are easily represented to the network through binary input units which take on the value *one* or *zero* respectively. The input representation is simple here, because there are only two possibilities. In natural language, however, the issue of representation is more complicated, since the input can take on many different configurations. However, while the representation methods used in CNLP are somewhat different, the underlying principle is the same: the data (some aspect of natural language here) is represented as activations to a set of input units. There are two major types of representation common in CNLP: *localist* and *distributed* (Sharkey and Ronan 1993, 16).

4.2.1 Localist representations

The localist representational scheme is the more straightforward; it entails using one computing element for each input entity. For natural language processing, the input entity might represent a word, letter or sound. For example, in the network designed to determine the gender of French nouns, the input units might represent the letters of the alphabet.

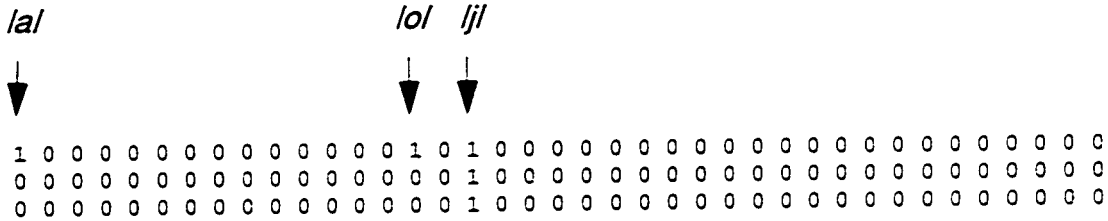
Once a localist representation scheme has been selected, there are still many ways of representing the input data. For instance, the gender selection problem might be resolved by coding inputs to represent phonemes instead of letters. In fact we constructed such a network as part of the preliminary work for this thesis, and it can be seen schematically in Figure 3.

Figure 3 The French gender network



Here, the input consists of 117 input units, taken in three groups of 39 units each. Each of the 39 units represents one French phoneme, and corresponding units are used to represent the number of repetitions of the phoneme within a word (to a maximum of three occurrences). For example, the input representing the sound /j/ would be set to *1* in each of the three groups for the word *jaugeage*. Consequently, *jaugeage* would be represented to the network as shown in Figure 4.

Figure 4 Jaugeage as represented to the neural network



Here, each column represents a different phoneme. The top row represents the first occurrence of the phoneme, the middle row represents the second occurrence of the phoneme, and the bottom row represents the third occurrence of the phoneme.[8] This approach might be preferable to one

8. One could also use integer-valued (rather than binary-valued) units to represent the phonemes in a word. The presence of a non-zero integer value at a given unit would indicate that the corresponding phoneme was present; the actual value of the integer would indicate how many times the phoneme appeared.

which takes the input units to represent letters of the alphabet, since words of any length can be represented as long as no phoneme appears more than three times. The output layer consists of two elements. One represents the masculine gender, and the other represents the feminine. Many different localist representations are available to the CNLP researcher. There is no rule for deciding a priori how to represent input; the *best* way is that which provides the output which best suits the researcher's purpose.

With an understanding of neural network training methods and the localist representation technique, we are now perhaps in a better position to appreciate the enigmatic nature of the functioning of neural networks. The adjustment of connection weights between units in a network creates a processing entity which mimics an aspect of the French language; namely that French nouns carry a gender. While the network's ability may be trivial (only a small subset of French nouns are accounted for), it admirably demonstrates the marked difference between traditional computing approaches and neural computation. From a traditional standpoint, general rules about the gender of words with certain endings would be developed, and then coded in software. No such rules are necessary for the network; it matches input to output by devising its own algorithm (for lack of a better word), which is somehow distributed across the network as a function of connection weights and network architecture. This fact is sometimes difficult to digest. The mechanics of *how* neural networks operate are well known, but *why* they work is another question. Hecht-Nielsen, a noted connectionist researcher, goes so far as to suggest that the answers to such questions "may require an intellectual revolution in information processing as profound as that in physics brought about by the Copenhagen interpretation of quantum mechanics" (Hecht-Nielsen 1990, 10).

4.2.2 Distributed representations

The other major type of representation is called *distributed*, since representations of this sort typically spread the representation of an entity over several input units. Within the field of CNLP, an example of distributed representation is the *microfeature*, or *vector* approach (Sharkey and Ronan 1993, 16). In this approach, *feature spaces* are constructed containing coordinates related to the meanings of a set of words (i.e., their semantic features). A small feature space might look like this: {*animal, living, big, pretty, walks, aggressive*}. Using a scale of -5 to +5 for each feature, the real-world concept "friendly-tame-dog" may then be represented as the vector {+5 +5 -1 +2 +5 -4} (Gallant 1993, 43). Redundancy in some of the features—for instance, something that walks is most likely animate—tends to make the network error-resistant. After all, an error in any one input (or output) may not be so serious if other units carry similar information. Feature sets more complicated than the one discussed here can be used for such tasks as the semantic analysis of natural language phrases. This approach shares much of the spirit of Nida's contrastive componential analysis, by which source language (SL) terms can be broken down into essential semantic features and these features compared to those of possible equivalents in the target language (TL). From Nida's perspective, reducing words to features is helpful both in understanding the SL text and in selecting a TL equivalent, since the latter may be chosen based on the degree to which it shares important features with the SL term. The notion of features also overlaps with Nida's opinions on connotative meaning, which he describes as "the meaning which deals with our emotional reactions to words" (Nida 1969, 91). Borrowing from the linguist C.E. Osgood, he shows how the connnotative meanings of the terms "mother" and "woman" might be measured using a scale of 1 to 10 on a set of contrasting pairs of adjectives such as "good/bad", "strong/weak", etc. Given this similarity of approach, we might be inclined to think such feature representations could be

adopted for connectionist machine translation. In fact, this approach has been proposed, as will be discussed in the section "Literature Review" below.

4.2.3 Hidden layers

Hidden layers are essential when networks are required to represent complex relationships between input and output. The layer develops its own interpretation of training input and this internal representation is used in providing correct network output. Intuitively one expects the extra layers of processing units will provide greater freedom in the way a network maps input to output, since more connections are available for weight adjustments and thus for representing the mapping function from input to output. Nevertheless, what the hidden layer actually learns is not easy to determine in many cases: "the cells become parts of distributed representations and are hard to interpret in simple terms," says Gallant (Gallant 1993, 225). Perhaps for this reason, a network is sometimes treated as a *black box* function in which the external world presents inputs to the input cells and collects output from the output cells without users understanding what goes on between the two. It is perhaps enough to say that the number of hidden layers and units is an important variable in network architecture, and one which is often determined, as already mentioned, by trial and error.

CHAPTER 5: LITERATURE REVIEW OF CONNECTIONIST MT

The first mention of translation in the connectionist literature appears in the proceedings of the First IEEE Conference on Neural Networks in 1987. As part of a preliminary investigation into the applicability of neural networks to natural language processing, Robert Allen examined some simple neural network configurations designed to translate a set of English sentences into Spanish (Allen 1987). He made several simplifying assumptions: sentences were computer-generated and contained only a subject, verb, direct object and indirect object; the verbs used were "to offer" and "to give"; the verbs appeared only in the present, past, or past perfect tenses. The English vocabulary amounted to 31 words, each encoded in 5-bit binary format (a localist representation). The Spanish vocabulary consisted of 40 terms coded in 6-bit binary format. The English sentence template accomodated 10 words, and shorter sentences were padded with null place-holding terms. A total of 3310 unique English sentences and their translations were generated using the above restrictions, and all but 33 were used as training data for the network (the remaining 33 sentence pairs were used as a testing set). In fact Allen tried three different network architectures, each trained using backpropagation. The most successful was a multi-hidden layer model, containing 50 input units, 66 output units, and 3 hidden layers of 150 units each. Allen reports that on average, this network translated 1.3 words per sentence incorrectly. Allen points out that even this result is remarkable, given that the network started with no semantic or syntactic information, and suggests that greater accuracy might be achieved by incorporating "multiple types of information" such as part-of-speech, gender or number information in the input and output layers. From this and the other studies in his article (including work on anaphora) he concludes that connectionist approaches to natural language processing are "quite promising". He points the way to further

research that might profitably be undertaken. For instance, he recommends that a network be designed to account for the sequential nature of language, i.e., the way the meaning of a sentence is built over time, from the utterance of the first word to the utterance of the last. We will discuss later the work of another researcher, J. Elman, who has recently investigated building meaning by presenting words sequentially to a neural network.

In a paper presented at the conference Theoretical and Methodological Issues in Machine Translation in 1993, Ian McLean took a slightly different route (McLean 1992). He suggested that a neural network could be used to calculate the *distance* between an SL phrase and its possible TL equivalents—the first step in the matching process of example-based machine translation (EBMT.) From an EBMT perspective, the TL equivalents considered closest to corresponding SL terms are used in constructing an appropriate TL sentence. McLean used a localist representation scheme for input with a 10 word maximum sentence length and a 30 word vocabulary. The input layer consisted of 300 units—each of the 10 word positions being associated with 30 units, one for each word in the vocabulary. The output layer contained a set of units each representing a "unique example identifier", i.e., a pointer to the closest TL sentence. The fully connected single-layer feedforward network was trained on 32 sentences from business letters, and several experiments were performed. MacLean found that the network was able to differentiate between words in the input pattern and make reasonably accurate generalizations about unseen input sentences (i.e., attach the same example identifier to similar sentences). He also found the network was able to utilize information on sentence length in differentiating similar source language sentences. Finally, MacLean determined that the network was also able to use frequency information to classify sentences, i.e., where two example identifiers were equally plausible by word length, the net-

work would opt for the one which occurred most frequently in the input corpus. He concluded that connectionist networks are "suitable for the matching process required in EBMT". He suggested that the levels of activation of units in the output layer offer a natural measure of the degree of confidence in the translation, an important criterion in determining sentence *closeness*. The one major problem with the system is its reliance on the absolute position of words within the input sentence. The network would see no similarity between the sentences "John walked through the door", and "Mr. Smith walked through the door", since it would try to match words position-for-position ("John" with "Mr." and "walked" with "Smith" and so on). Another problem is establishing a plausible and exhaustive set of prototype sentences with which a wide variety of input sentences might be aligned, and establishing criteria by which one sentence would be considered "near" to another. Also, the similarity of one sentence to another must be determined by non-connectionist means before the network can be trained. Of course these a priori matches need not be exhaustive if the network can successfully generalize from a small set of previously categorized sentences.

While very restricted in scope, McLean's effort is interesting because he used phrases taken from everyday language (business letters in this case). This at least offers hope that such a system might be applied to real-world problems.

In a 1991 paper called "A Practical Approach for Representing Context and for Performing Word Sense Disambiguation Using Neural Networks" (Gallant 1991), Stephen Gallant suggested a method by which a full-scale natural language processing system might be implemented. Gallant proposed to use feature spaces and context vectors to accomplish word sense disambiguation. As already described (page 27), a feature space is a set of concepts which can be used in combination

to define any word in a given corpus. Each word is represented as a vector of numerical values each of which represents the degree to which it corresponds to a certain element in the feature space. Gallant does not propose an exhaustive feature space, but suggests that it might include such concepts as "human", "man", "woman", "machine", "politics", "art", "science", "play", "sex", "entertainment", "walk", "lie-down", "motion", "speak", etc. He suggests that context vector values fall between -2 and +2. The feature space would vary to reflect the kind of words it would be required to disambiguate. Each word in a text, then, becomes a vector of numbers, each reflecting the strength of association between the word and a feature space element. For instance, the context vector for "astronomer", assuming the feature space above, would be {+2, +1, +1, -1, -1, 0, +2, 0, 0,...}. Once the context vectors for a set of words have been established—Gallant estimates this could be done manually for a 2000 word corpus, using 200 feature space elements,[1] in 80 person-days—they can be used to determine the most reasonable meaning for a given word in a sentence. To do this, he proposes calculating a *dynamic context vector* at each position in a sentence. The dynamic context vector is calculated at each word in a sentence by a *context algorithm*, which may take into account context vectors from the current sentence and from previous sentences. The exact contribution of these context vectors to the current dynamic context vector depends on the context algorithm being used. Gallant proposes two context algorithms in his paper, but the effect is the same in both—the influence of previous words "decays gracefully with distance" (ibid., 298). Gallant proposes implementing the approach on a neural network as follows: the network input is the dynamic context vector at the point in a sentence where the meaning of the next word must be chosen. The output layer of the network consists of units representing competing meanings for a word (each meaning represented by a single unit), with the most likely carrying the

1. Gallant gives no evidence to support the contention that 200 elements would be sufficient.

highest activation. The network could be trained with training pairs consisting of dynamic context vectors for input and appropriate meaning selections as output. Gallant has not actually implemented his model; his stated purpose is to encourage a "large-scale implementation of the context vector approach" (ibid., 294).

The two main practical applications suggested by Gallant are MT and Japanese word processing. He believes his approach would improve the accuracy of word choice in MT systems, reducing human editing requirements and improving throughput. In Japanese word processing, Gallant's system would provide the context necessary to correctly determine the appropriate alphabetical equivalent of kanji characters.

Such a system has obvious potential, but drawbacks as well. First, comes the problem of knowledge representation. What constitutes a good feature set? Are there universal axioms suitable for categorizing the meanings of words? Can they be captured? The premise is that word meanings can be classified by sets of meta-words representing axiomatic idea elements, but this is far from certain. In fact there is the scent of infinite recursion in such an approach, since words or word-like elements must always be used to classify other, more specific words. Is "human" really elemental, or could it be advantageous to break it down further into a set of defining characteristics. Should another collection of words be used to describe each organ, and so on? These questions are currently being considered by many AI researchers, but their work is beyond the scope of this thesis. The other problem is assigning context vectors to word meanings in an appropriate fashion. Exactly how human is a chimpanzee that can count? Gallant feels that given a feature space containing sufficiently redundant elements, it is enough to assign the context vectors in a "reasonable" fashion. System performance will not depend uniquely on the accuracy of a particular vector

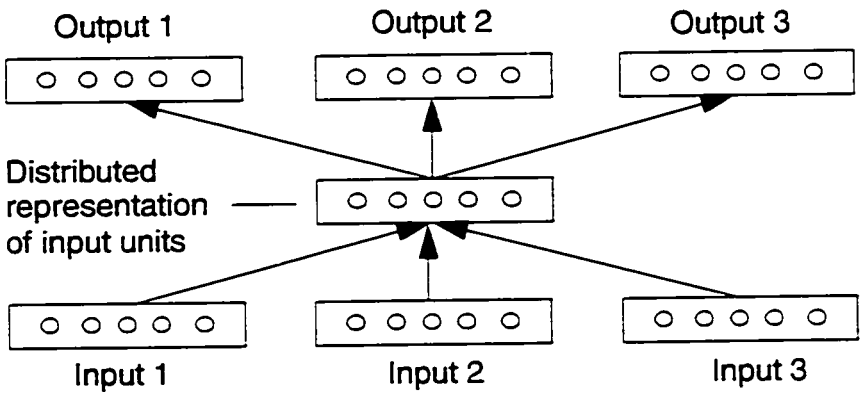
element, since similar redundant vector elements are likely to be correctly coded. Overall then, the context vector for a given word will be adequate.

The methods mentioned so far entail the use of backpropagation networks, but other types of networks have also been used for natural language processing and translation. They arose in large part because of a now famous paper by Fodor and Pylyshyn (1988) condemning the inability of some backpropagation networks to account for the combinatorial nature of language (a characteristic they called *compositionality*). They contended that neural networks would not provide a realistic model for language processing until systems were developed that could represent serially ordered inputs. The first to respond to the challenge was Pollack (1988) who developed a connectionist framework using an auto-associative network[2] that was able to encode the compositional structure of language. He called his system RAAM (Recursive Auto-Associative Memory), and his ideas were convincingly applied by Chalmers in 1990 to the task of sentence passivization. The RAAM architecture used by Chalmers is shown in Figure 5, where the input and output units represent sentence constituents. In simple sentences, the sentence constituents would be words. For complex sentences with more than one clause, the sentence constituents would be both words and sentence fragments, as explained below.

---

2. An *auto-associative* network is one that has been trained to reproduce at its output layer the values presented to its input layer (cf. p.14).

Figure 5 The RAAM architecture



The main feature of the RAAM is its ability to develop a distributed representation of an input vector at its hidden layer. This is accomplished by training the network to auto-associate its input units (that is, training the network to match output vectors to input vectors). Since the input and output vectors are the same for any given training pair, the set of activations at the hidden layer is effectively a distributed representation of the input vector. One can create a distributed representation of a set of input vectors in the following way: present an input vector to the input layer and propagate it to the hidden layer; feed the hidden layer activations back to the input layer and introduce a new input vector; propagate the input vector and activation values to the hidden layer; feed the hidden layer activations back to the input layer; introduce a new input vector; and so on until all the input vectors in the set have been propagated to the hidden layer. At that point the hidden layer effectively contains a distributed (and compressed) representation of all the input vectors presented to the network.

Chalmers used this principle to passivize English sentences. There were three steps in his experiment. First, he automatically parsed his sentences into syntactic trees with a maximum of three

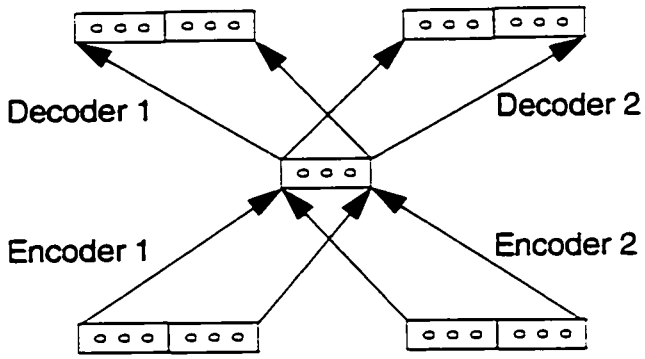
leaf nodes to each branch. Starting at the bottom of the tree, he fed three leaf words into the network and propagated them to the hidden layer. The hidden layer thus became a distributed representation of that branch of the sentence's syntactic tree. He then copied the activations of the elements in the hidden layer back to elements in the input layer, introduced fresh input vectors that represented other leaves or branches of the tree, propagated these to the hidden layer, and proceeded in this manner until the hidden layer activations provided a distributed representation of the entire tree. He repeated this process to produce distributed representations of 80 active-passive sentence pairs. The next step was to use forty of these distributed representations to train a two-layer feedforward transformation network. The purpose of this network was to convert active sentences into their passive form; for instance, "John loves Michael" would be converted by the transformation network to read "Michael is loved by John" (Chalmers 1990, 54). The distributed representation of the active sentences was presented to the input layer of this second network, and the distributed representations of the corresponding passive sentences were presented to the output layer. Once this network had been trained, Chalmers presented it with each of the 40 active sentences in the test set he produced using the RAAM. The transformation network produced distributed representations of the corresponding passivized sentences at its output layer. The third and final step in Chalmers' experiment was to present the output from this transformation network back to the hidden layer of the RAAM. By reversing the recursive process by which he built distributed representations of sentences, Chalmers was able to decode the representations created by his transformation network.

Using this approach, Chalmers was able to correctly passivize 40 short active sentences with 100% accuracy. Nonetheless, Chalmer's experiment was limited in scope, using as it did a 13

word vocabulary and allowing only two single-clause syntactic structures. The sentences were encoded using an augmented localist representation in which each word consisted of 13 units, six for part of speech, five for the word itself, and two unused. The sentences were randomly generated by computer.

This general approach was taken up by Chrisman (1991). Chrisman's modification of the RAAM framework did away with the necessity for a separate transformation network: the transformation task was combined with the task of creating a distributed representation of the input sentence. He called his network a dual-ported RAAM. While the network could be used to perform any number of inference tasks, Chrisman decided to use it for natural language translation. His dual-ported RAAM can be thought of as a 2-layer auto-associative network consisting of two *encoders* and two *decoders*, as shown in Figure 6.

Figure 6 The dual-ported RAAM architecture



Each encoder is a set of input units, and for Chrisman's translation experiment one set was reserved for an English phrase and the other for its Spanish equivalent. The decoders are two sets

of output units: one for the English phrase and the other for its Spanish equivalent. This allowed the network to translate in both directions. Sentences were input recursively following Chalmer's model, and the network trained so that each complete representation of an English sentence produced the same English representation on one decoder and an encoded representation of its Spanish equivalent on the other decoder. He used 216 English-Spanish sentence pairs constructed from 36 English and 36 Spanish words. The English words were encoded using a 22-bit scheme, 9 bits for part-of-speech, 3 for plurality and 10 to identify the word itself. Interestingly, his scheme allows for words sharing the same root to be coded with the same identity bit, establishing an explicit link between related words such as "professor" and "professors," and "do" and "does." This attempts to address one of the problems of strict localist representations in which words of different spellings are given unique representations regardless of their kinship to other words. Localist representations make for very large vocabularies in full-scale systems, since they prohibit the natural economy provided by simple morphological analysis.

Chrisman successfully trained the network to relate all 216 English sentences to their Spanish equivalents, with a total of nine words weakly learned. A word was weakly learned if the value of one of the elements in its output representation fell between 0.2 and 0.8. He reported finding no explicit relationship between hidden layer activations and specific features of the input data. In another experiment he tested the generalizing capabilities of the network, training it on 108 sentence pairs, and using the remaining 108 pairs for testing. Two types of error were possible in the system—the RAMM might incorrectly encode an unseen sentence, or it might mis-translate a sentence. In Chrisman's experiment, the network was successful in encoding 87% of the test sentences, and in translating 75% of them. Only translations that exactly matched the established

equivalent were considered correct. Of the incorrect translations, 87% differed from the equivalent either by a single word or a subject with consistent verb conjugation. When encoding errors were eliminated, the network translated 89% of the sentences correctly. 60% of the mis-translated sentences differed from their target representations by a single word or by an "incorrect subject with a verb agreeing in conjugation" (ibid., p. 357).

While this result is encouraging, Chrisman takes care to point out that the result may not extend to natural language translation in general, since his dataset is small and not necessarily representative of natural language in the wider sense. He concludes that the system is able to extract "deep semantic and/or syntactic commonalities between an input problem and its answer" (Chrisman 1991, 359), although one can argue that this is overstating the fact. The system's only *semantic* cues are the words themselves, and these are represented as *on* or *off* bits in a 10-bit word. The system establishes a relationship between patterns of bit activity and inactivity in the input and output layers. It is our interpretation that imbues the activation patterns with meaning. Chrisman also makes an interesting connection between the intermediate representation created by his method in the hidden layer and the type of interlingual representation that has long been discussed in machine translation circles (Sadler and Wikkam 1986). He rightly points out that no intermediate language of sufficient representational power has been found for the approach. Using Chrisman's technique, however, an interlingual representation emerges as a natural product of the approach. Again the caveat applies that the vocabulary and syntactic constructions used in Chrisman's work were much more restricted than those useful in any machine translation system. An interlingua could almost certainly be constructed to correctly translate all the sentences in his training sequence.

Another method of accounting for the sequential nature of language using neural networks was implemented by Jeffrey Elman, building on work by Jordan (1986). Elman, like Pollack, copied the activations of hidden units produced by one input vector back to the input layer to enhance the input vector that followed. This allowed the prior state of the network to influence the current state. As training proceeds, the hidden units develop a representation that encodes some aspect of the sequence of inputs that preceded the current input. While Elman did not apply this approach to translation, it is an interesting one because it could be applied to the task, at least if one assumes that the words surrounding a given SL term have some role in conditioning the selection of its equivalent. In Elman's network, values supplied to the input layer were propagated to the hidden layer. The activations of the hidden layer were then copied to a *context layer*, which is then combined with new input values to produce an input vector that takes into consideration the previous input. On the next cycle, the hidden layer activations could be said to have encoded in some fashion the distillation of the previous input vectors (in the same way that Pollack's RAAM develops a distributed representation of sets of input vectors). To test his theory, Elman generated 10,000 two- and three-word sentences from a lexicon of 29 nouns and verbs. The sentences were based on templates that reflected certain properties of the words—for instance, intransitive verbs carried no objects, only animate nouns occurred as the subject of the verb "to eat", etc. The words were represented locally, and provided to the network input one at a time. The network was trained to represent the next word in the sequence on its outputs; the task for the network was therefore to predict the next word in a sequence given information on the words that preceded it. Elman compared the output with the expected frequencies with which a possible successor would occur (as extracted from the training corpus), and found that the network readily learned to approximate the likelihood of a given word occurring next. Elman then performed a cluster analysis of the repre-

sentations occurring on the hidden layer at the introduction of each word. Based on the similarity of their hidden-layer representations, the words fell into several categories: one corresponding to verbs, one to nouns, and several subcategories containing verbs that require a direct object, verbs that do not, nouns that are animate, those that are inanimate, etc. The network had discovered lexical classes from word order. He points out that the network "recognizes that (in this corpus) there is a class of inputs (namely, verbs) that typically follow other inputs (namely, nouns)". He also notes that the network has no semantic concept of what is contained in the categories, since it has no information to relate them with objects in the real world. Another researcher, Jay McClelland, suggested that the feat was similar to learning a "language by listening to the radio" (Rumelhart and McClelland 1986, 210). As previously mentioned, Elman then introduced an imaginary word into his corpus whose binary coding differed from all of the words in the training data. He dubbed the word "zog", although any name would do. When presented with sentences containing the word "zog" in place of "man", the already trained network assigned it an internal representation that placed it in with the other animate nouns in the corpus. Elman proposes this as a partial explanation of how human language learners use knowledge of how a word is used to determine the approximate meaning or class of a new word. Of course this result would be expected, since the network is oblivious to the content of its categories—only the way a word is used (as evidenced by its position in a sequence) figures in its categorization process.

Stephan Wermter (1994) has recently investigated an Elman-type recurrent network for classifying the titles of articles. His work interests us here because the network he constructed works equally well on English and German titles, and because context was a determining factor in arriving at final classifications. He collected titles in three categories (computer science (CS), history/

politics (HP), and materials/geology (MG)) from the on-line catalog of a university library. Each word in a title was converted to a *significance vector* consisting of three numbers representing the relative frequency of the word in each class. For instance, the word "processing" had a significance vector of <0.00,0.94,0.06> (HP: 0.00, CS: 0.94 and MG: 0.06). This vector indicates that, as might be expected, the word "processing" appeared much more frequently in computer science titles than in either history/politics or materials/geology. Wermter designed a recurrent two-layer network with context and hidden layers of three units each. The significance vectors for the words in a title were presented one after another and the network trained to classify the title once all vectors had been input. The context layer provided a kind of short-term memory to the network, allowing the words in a title to cumulatively decide its classification. He then tested the network using 286 new English and German titles. His results were very positive: the network classification agreed with that of the library in 99.6% of cases. Thus the network successfully generalized from the training set to the testing set. His results also showed that class assignment was dynamic throughout the presentation of input vectors, i.e., classifications sometimes changed as new vectors were presented to the network. This approach is interesting from a translation perspective because it makes use of contextual information to dynamically assign classes. Context is very likely an important factor in determining the appropriate equivalent for an ambiguous SL term, and we will hopefully be able to exploit this feature in a connectionist approach to translation. One note of caution regarding Wermter's procedure: he extracts word frequency data from his corpus and provides this to the network. In other systems, the network is left to infer the word frequency regularities on its own. With this step removed, one wonders whether a neural network was truly necessary to determine the classification. It would be interesting to see how a simpler technique such as vector addition would fare in comparison.

The last translation-related research we will examine was conducted by Kazuhiro Kimura and reported in the Proceedings of the 30th Conference of the Association for Computational Linguistics, 1993. His special concern is translating Japanese phonetic representations (kana) into the complex written form called kanji. The problem is a difficult one, since a given spoken word can have several meanings, and thus can be represented by several different kanji characters. Kana-kanji conversion is currently a two-step process: words are input in kana and morphologically analysed before conversion. Any ambiguities (of which there are many) are then passed to a homonym selector, which uses co-occurrence rules, selectional restrictions and a context bank to resolve them. Nevertheless, the degree of polysemy is so great that the final choice of kanji equivalents is often left to a human monitor. Kimura proposes using an associative neural network to enhance the ability of the system to select the appropriate kanji character. Kimura gives the network the role of ranking the possible kanji equivalents for a given kana input based on the kanji characters that have come before in the text. The associative network contains a node for every kanji character and is trained using real-world documents from a given domain. The co-occurrence frequency of kanji characters in a paragraph become the connection weights between nodes representing them in the network. At this point, the network is ready to help disambiguate kana entries. As the network user enters kana text, network units representing kanji characters are activated. As is the case with associative networks, these activations spread to connected units in the network. When the next polysemous kana word is encountered, the units representing possible kanji equivalents are polled for their activation levels. The likelihood of a given kanji character being the correct one is proportional to the activation level of its corresponding unit. The kanji characters are thus ranked according to their activation level in the network, and the user is given the final choice. Kimura constructed four different networks using text from four fields: business

letters, personal letters, news articles, and technical articles.[3] He found that the neural network selector reduced the number of choices that had to be made by the user in all four text categories by an average of 8.33% more than the reduction afforded by the use of conventional kana-kanji converters without a neural network extension. Kimura's work is interesting for three reasons: first, it offers another approach for capturing contextual information and using it to disambiguate polysemous terms; second, it makes use of real-world documents for training—one of the few instances of such a practical approach; and third, it takes advantage of co-occurrence consistency within broad categories of sub-languages, namely business letters, personal letters, new articles and technical articles. The approach to translation taken in this thesis incorporates all three of these aspects.

Overall, given the relative youth of neurocomputing, a surprising variety of approaches has been used to attack natural language translation. Nonetheless, on first sight at least two significant areas of investigation have been overlooked. First, no language-to-language network has been trained on a real-world corpus of any size, and second, none have been developed specifically to operate on domain-specific texts (thereby restricting vocabulary size naturally). The network developed as part of this thesis attempts to address both oversights.

3. Note that these categories, particularly personal letters and technical articles, are very broad.

CHAPTER 6: THE EXPERIMENT

This section describes TransNet, a neural network designed to translate natural language weather reports from English into French. TransNet was designed, trained, and tested by the author in conjunction with suggestions made by his thesis advisors.

6.1 Objectives

The purpose of this thesis is to explore the applicability of neural networks to natural language translation. We felt that building a neural network ourselves was an important part of accomplishing this task. We had two major objectives in building the network, both of which extend upon existing work in the field. These were:

- to train a network using a relatively large natural language corpus with a suitably interesting vocabulary
- to account for word order in the data representation for input to the network (instead of in the network structure)

As previously mentioned, most of the attempts at translating natural language using a neural network have involved very small vocabularies. For instance, Chrisman (1991) used a total vocabulary of 36 English and 36 Spanish words in his experiments with natural language translation. From this vocabulary he constructed (by computer) 216 sentence pairs. We felt it would be interesting to use a larger number of sentence pairs, and collect those sentence pairs from naturally occurring text, the idea being that the network output would then resemble translations produced by humans. This implied use of a fairly large corpus of bilingual text. Nevertheless, we knew that

it would be wise to constrain the corpus vocabulary as much as possible. The simplest way to do this was to use a sublanguage corpus,[1] thus restricting the vocabulary naturally.

Secondly, much of the recent work in natural language translation using a neural network has focused on using the network structure to account for the word sequences (for example, Chalmers, 1990 and Chrisman, 1991). However, since our expertise was in translation rather than network construction we decided to maintain a simple network structure and instead put the burden of representing word order on the data representation.

6.2 Methodology

We used a software package called the Xerion Neural Network Simulator (Version 4.1), to build, train, and test TransNet. Xerion was designed by the Connectionist Research Group at the University of Toronto. It is available at no charge to Canadian industry and researchers. Xerion consists of a set of C libraries that can be used to build neural networks, and several pre-built simulators that make use of the libraries. It uses the X-Windows graphical interface and provides an intuitive specification language for building networks. It also has the advantage of extensive documentation, including over 100 UNIX help pages and a tutorial/reference manual. We ran Xerion on a variety of machines including an HP712 with 96 MB of RAM (running HPUX 9.03) and a Sun SparcStation with 64 MB of RAM (running SunOS 4.1). The network we built used the Xerion back-propagation simulator for training.

There were six distinct phases in building the network. They were: deciding on a corpus for training and testing the network, compiling the corpus, pre-editing the corpus, designing a data repre-

1. *Sublanguage* in the sense given to the term by Kittredge et al. (Kittredge and Lehrberger 1982). That is to say, the language of a certain genre of document in a very specific technical field.

sentation, converting the corpus data, and building and running the neural network. The next section describes each step in detail.

6.2.1 Deciding on a corpus

Choosing a corpus was the first obstacle to creating a neural network translation system. Since neural networks need to be trained, we needed a large amount of bilingual translated text paired sentence by sentence (sometimes referred to as *bitext* (Harris 1988)). The neural network would be learning from this bitext, so we also had to make sure that the translations it contained were relatively consistent. The more consistent the translation, the less noise (i.e., alternative translations) the network would encounter when it tried to extract translation rules. The best way to achieve consistency was to use text taken from a particular subject domain, as this would help reduce the amount of ambiguity in the corpus and thus simplify the task to be presented to the network. While in theory there are many possible translations for a given utterance, in practice a few almost always prevail in specific domains. For example, while the text segment "Showers today" could be translated into French a number of ways (e.g., "Allez prendre votre douche"), the prevailing translation in weather reports is "Pluie aujourd'hui". By selecting the language of weather reports as our sublanguage and presenting only the latter translation to the network, we spare it the difficulty of attempting to infer rules for all possible translations at once. Another limitation was vocabulary size: we knew that regardless of the data representation we chose, computer memory and processing power limitations would likely prevent us from representing a large number of words to the network. Luckily a restricted vocabulary is one of the characteristics of a sublanguage. In all then, we identified four criteria for a corpus. It had to:

- be large and bilingual and already aligned sentence by sentence

- consist of naturally occurring sentences or sentence fragments (not artificially generated ones)
- be readily available
- be restricted to a controlled vocabulary (preferably as a natural consequence of the domain it describes)

In short, we were looking for large amounts of sublanguage bitext. Perhaps the best-known and best-documented sublanguage in the field of MT is the one used in Environment Canada weather forecasts.[2] Environment Canada always provides its weather reports in English and French and a large segment of them is translated by a machine translation system called METEO (Chandioux 1976). We were not able to track down an existing electronic bitext corpus in this field, so we had to build one. Fortunately, our task was made easier by Environment Canada's decision to make its bilingual weather reports available on the World Wide Web.

6.2.2 Compiling the corpus

Environment Canada posts weather reports to its Web site several times a day. Currently, the information is available at "http://www.on.doe.ca/text/". To build the corpus, we collected English and French weather forecasts from the Environment Canada Web site for one month, up to three times a day from 16/05/95 to 16/06/95. The forecasts are accessible by region, and we chose reports from four regions in Quebec: Quebec North, Quebec South, Quebec East, and Quebec West. During the one-month period, we collected a total of 176 forecasts (88 in each language), occupying 759,743 bytes of disk space. The total number of sentences collected was 6,478 (3,239 sentences in each language). When combined, this corpus became a 410-page WordPerfect file.

---

2. The choice of domain was made in 1992. Since then much more electronic text has become available over the Internet. Were we choosing the domain today, we might choose a different one.

After conditioning the corpus (this process is described in the next section), we discovered that it contained 143 English word types and 146 French word types.[3]

We later discovered that many of the sentences were repeated extensively throughout the corpus. Not counting repeated sentences, our corpus contained a total of 1,782 sentences (that is, 891 English sentences, and their 891 French counterparts).

6.2.3 Pre-editing the corpus

Each weather forecast contained header information including a report number, location, and time of issue. Figure 7 shows an example of an unedited weather report. Both English and French reports followed an identical format.

Figure 7 Raw METEO output[4]

```
FPCN13 CWUL 230811
FORECAST FOR CENTRAL QUEBEC ISSUED BY ENVIRONMENT CANADA
MONTREAL AT 05.00 AM EDT TUESDAY MAY 23 1995 FOR TODAY AND
WEDNESDAY.
NEXT FORECAST ISSUED AT 12.00 PM EDT.

TROIS-RIVIERES-AND-DRUMMONDVILLE.
TODAY.. INCREASING CLOUDINESS. RAIN BEGINNING LATE THIS
AFTERNOON. HIGH NEAR 18. WINDS 15 TO 30 KM/H BECOMING
SOUTHWESTERLY 30 TO 50 THIS AFTERNOON.
TONIGHT.. RAIN. LOW NEAR 9. WINDS 20 TO 40 KM/H.
WEDNESDAY.. RAIN. RISK OF THUNDERSHOWER. HIGH NEAR 17. WINDS 20
TO 40 KM/H.
```

3. A word *type* is one specific word. Each occurrence of that type is called a *token* (Hartmann and Stork 1972).
4. The assumption here is that the translations have been done by METEO, although we have no means of verifying this.

To make the reports machine-readable for our network, we had alter the format shown in Figure 7. To do this we deleted all the headers and put each sentence on its own line ending with a period. In all cases, the English and French forecasts contained the same number of sentences.

We regularized the terminology and the orthography. We changed "acres w" (obviously a typo) to "apres-midi"; "jusqu'a" to "jusqu a"; "aujourd hui" to "aujourdhui"; "jusqu tot" to "jusqu a tot". We also replaced all numbers with the *placeholder* word "NUM" and names of days with "DAY-LABEL." The placeholder words helped limit the size of the vocabulary.[5] Placeholders were only used when translation of a word could always be done by simple one-for-one substitution: for instance, the number "1" is always translated by the number "1" in weather reports. The following spelling corrections were made: "endrois" to "endroit"; "hundershower" to "thundershower"; "l" to "le"; "vens" to "vents"; "bu" to "by"; "km/l" to "km/h"; "cent" to "pourcent". In some cases, a French word had been inserted in the English version of a weather report. We repaired these instances too: we changed "vens" to "winds"; "voile" to "variable"; "by endroits" to "locally"; and "minres" to "low near".

Also, we lemmatized contractions of prepositions and articles: "d'" to "de"; "l'" to "le". "Du" became "de le"; "des" became "de les"; "au" became "a le".

After the above changes were made, we were left with a clean corpus of 1,782 sentences. See Figure 8 for an example. Pre-editing our corpus was time-consuming; however, we wrote wordprocessor macros to simplify the task and these might be integrated into a fully automated pre-editing system.

---

5. On the downside, using placeholders also increases the ambiguity of sentences that contain them. However in this case we were willing to make the tradeoff between ambiguity and vocabulary size.

Figure 8 Pre-edited METEO text

```
THIS EVENING AND TONIGHT CLOUDY PERIODS.
LOW NUM TO NUM.
DAYLABEL SUNNY WITH CLOUDY PERIODS.
HIGH NEAR NUM.
DAYLABEL CLOUDY WITH CLEAR PERIODS AND NUM PERCENT PROBABILITY OF SHOWERS.
LOW NEAR NUM.
```

6.2.4 Designing the data representation

One of the stated intentions of this thesis was to put the burden of representing word order on the data rather than the network. An example of building word order into the network itself is the work of Chalmers in 1990. Chalmers built a feedback loop into his neural network so that each input word had an influence on the word or words chosen to translate subsequent words (see above, page 34.) However, this approach has the drawback of complicating the network structure. We felt it would be interesting to try an approach that entailed building word order information into the data presented to the network. To our knowledge we are the first to have tried this method. Designing a representation for individual words without taking word sequence into account is relatively simple. One can simply reserve a single input unit for each word in the source corpus and a single output unit for each word in the target corpus. If the word appears in the input sentence, the corresponding unit in the input layer is activated, otherwise it is de-activated. The input representation for a sentence would be a collection of activated and de-activated units but without any information about word order within the sentence. The output is also a collection of activated and de-activated units without word-order information—an unordered set of words. However, there is not much demand for translating unordered lists of words from a source

language into unordered lists of words in a target language, so we decided to build a data representation that could account for word order.

Inspired by the research at the IBM Thomas J. Watson Research Center into speech-to-text transcription software for dictation systems (Brown et al. 1994), we decided to use overlapping word triplets (sometimes called *trigrams*) as the unit of translation for our neural network. In designing their computerized dictation software, IBM researchers found they could often disambiguate an utterance by examining the elements it comprised in groups of three. For instance, in the utterance "The time is right, Mr. Wright", the IBM system would distinguish between the two homonyms "right" and "Wright" by examining the elements that occurred immediately to their left and right ("is" and "Mr." for the former, and "Mr." and "end of sentence" for the latter). For the speech researchers, the trigrams created a context for each word in an utterance which helped disambiguate that word. We decided to adapt this approach to our problem so that the elements in our network's input and output layers represented trigrams rather than words. For written text, taking words in the sentence in overlapping groups of three encodes local word order. We felt that this approach might also provide a very limited translation context useful for disambiguating words in the source language. Since every trigram except the first and the last overlaps with two others in the sentence (the one that precedes it and the one that follows it), it is also possible, in most cases, to reconstruct a sentence given the set of trigrams it comprises. Even the first and last trigrams overlap with one other trigram. Effectively then, this data representation provides us with sentential word order information.

This meant that our weather reports would have to be pre-edited further, this time to represent each sentence as a set of trigrams. We processed each sentence in our corpus to create a dictionary

of trigrams. Our corpus contained 1,276 unique English trigrams, and 1,426 unique French trigrams. We decided that it was best to dedicate a single input node to each English trigram, and a single output node to each French trigram. (The network translates from English into French). It thus followed that our network would contain 1,276 input nodes and 1,426 output nodes. Since our rule of thumb was always to start with the simplest network possible, we decided to avoid hidden layers for our first trial. Unfortunately, after developing this data representation, converting each sentence in our corpus into a set of trigrams, and building a 1,276 x 1,426 network, we discovered that our hardware and software were not up to the task. Just constructing a network of this size with the software we were using consumed more memory resources than were available on our workstation.[6]

Limited by our equipment, therefore, we decided to scale back our effort while retaining the spirit of a trigram representation. We chose instead to use word pairs (referred to here as *bigrams*), rather than trigrams as the unit of translation. Thus we re-converted the corpus sentences, but this time into sets of bigrams. In our corpus we discovered 672 unique English bigrams and 714 unique French bigrams. We then proceeded to build, train, and test the network.

Unfortunately, Xerion, our neural network simulation environment, again required more memory than was available. However, our computer was nearly able to build the entire network. Encouraged by this trend, we decided to reduce the scale of the project still further, and extract bigrams from only the first 800 sentences of the sentences in the corpus (400 English sentences and their

6. Xerion was designed to favour usability. We noticed in some cases that efficiency was sacrificed in consequence.

400 French translations). We discovered 506 French bigrams and 465 English bigrams in the newly constrained corpus. This is the corpus we used for the remainder of the project.

We decided to run 10 training and testing sessions.[7] In each of these, 200 sentence pairs (of the available 400 sentence pairs in the corpus) were used for training and the other 200 sentence pairs were used for testing. Each run used a different set of training and testing sentence pairs from the corpus. A minimum of 10 results was needed to ensure statistical significance.

Once we had established a training and testing plan and decided on the bigram representation, we had to convert the corpus sentence pairs into sets of bigrams and then split the resulting bigram corpus into 10 different training and testing sets.

6.2.5 Converting the corpus data

Our pre-edited and restricted corpus contained 400 English sentences and their 400 French translations. The first step in building our 10 training and testing sets, as we have said, was to convert the sentences in the corpus into sets of bigrams. The process was the same for both the French and the English sentences.

First, we added marker nodes to the beginning (BG) and end (FN) of each sentence. The purpose of these flags was to ensure that the data contained information on where the sentence began and ended, and at the same time to add a dummy word at the beginning and end so that each word token in the sentence participated in two bigrams. The location of a word with respect to other words in the sentence can only be known with certainty if it occurs in two bigrams. Thus, in the example below, the word "THIS" appears in the bigrams "BG THIS" and "THIS EVENING". We

7. A minimum of 10 results was needed to ensure statistical significance.

wrote an awk script called "addBegEndNode.awk" to automate the task of adding the BG/FN flags to the corpus. The result is shown in Figure 9.

Figure 9 Pre-edited METEO text with marker nodes (output from addBegEndNode.awk)

```
BG THIS EVENING AND TONIGHT CLOUDY PERIODS FN.
BG LOW NUM TO NUM FN.
BG DAYLABEL SUNNY WITH CLOUDY PERIODS FN.
```

The second step was to create a list of the bigram tokens in the 400 English sentences, and a similar list for the 400 French sentences. We wrote an awk script called "findbigrams.awk" to accomplish this task. This script converted all the sentences in the corpus into sets of bigrams. It saved the bigrams in two files, one for English bigrams and the other for French bigrams. This script used a special boundary string "BNDRY" to indicate sentence boundaries. Figure 10 shows an example.

Figure 10 Source sentences in bigram form (output from "findbds.awk")

```
BG THIS
THIS EVENING
EVENING AND
AND TONIGHT
TONIGHT CLOUDY
CLOUDY PERIODS
PERIODS FN.
BNDRY
BG LOW
LOW NUM
NUM TO
TO NUM
NUM FN.
BNDRY
```

The third step was to generate a list of all the bigram types in the 400 English sentences, and all the bigram types in the 400 French sentences. This was done by further processing the files cre-

ated in the previous step. We ran these two files through the UNIX *sort* and *uniq* utilities to eliminate duplicates. The result was two lists of unique bigrams (one English and one French). These were the dictionaries used to convert the text sentences into the binary representation demanded by the neural network. Some entries in the English bigram dictionary are shown in Figure 11.

Figure 11 Some entries in the English bigram dictionary (file "edtdct.bd")

```
A FEW
A HEAVY
A NUM
A RISK
A THUNDERSHOWER
ABITIBI REGION
AFTERNOON AND
AFTERNOON FN.
```

The final step was to convert the bigram sentences into a form recognized by our neural network. Since the network was set up to translate from English into French, the input layer would contain English bigrams, and the output layer would contain French bigrams. Further, we knew that there were 465 unique English bigrams and 506 unique French bigrams. Since we had decided to reserve a single network element for each bigram, each input (English) sentence would have to be represented by 465 values (one for each network element), each representing a single English bigram. An element would be assigned the value "0" if its bigram were not present in the sentence, or the value "1" if the bigram were present. Likewise, each output (French) sentence would consist of 506 values. Again, the value of a given element would be equal to "0" if the bigram were not present in the sentence, or "1" if the bigram were present.[8] We wrote a C program (called "bgfn_vrt.c") to automate this task. This program looked at each bigram in the sentence

8. A discussion of the limitations of this representation (such as the difficulty of representing more than one instance of a bigram in a sentence) is provided in the section "Reconstructing sentences" on page 73.

and checked to see where that bigram occurred in the bigram dictionary. If the bigram appeared in the first position in the dictionary, then the value of the first network element was set to "1"; otherwise the value of the first network element was set to "0". If the bigram appeared in the second position in the dictionary, then the value of the second network element was set to "1"; otherwise the value was set to "0". The process continued until all the bigrams in the sentence had been found in the dictionary. The program then saved the resulting binary representation to a file and proceeded to the next sentence. The program also enclosed the representations for each sentence in curly brackets as required by Xerion. An example binary representation produced by the program is shown in Figure 12. This is the binary representation for the sentence "This evening and tonight cloudy periods".

Figure 12 Input sentence in encoded bigram form for "This evening and tonight cloudy periods"

```
{ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 }
```

In Figure 12 there are 465 position; seven of these are set to "1". From this we know that there are seven bigrams in the sentence. They are "BG THIS", "THIS EVENING", "EVENING AND",

"AND TONIGHT", "TONIGHT CLOUDY", "CLOUDY PERIODS", "PERIODS FN". All of the other bigrams are set to "0", so we know they are not present in the training sentence. The program converted every sentence in the corpus into its equivalent bigram representation. Since Xerion requires that the input and output binary representations for each training or testing pair be adjacent in the same file (that is, it requires the format "{<English sentence binary representation>}{<French sentence binary representation>}", we wrote a UNIX script to combine the English and French representations into a single file.

The next step was to generate 10 training and testing sets from the corpus. We created one evaluation set using the first 200 sentences for training and the last 200 sentences for testing. We then created 9 more training and testing sets by choosing sentence pairs at random from the 400 sentence pairs in the corpus. We wrote a C program to generate 200 random numbers and then applied the numbers to select the corresponding sentences from the corpus using an awk script. Once 200 sentence pairs had been chosen for the training set, the remaining sentence pairs were placed in the testing set.

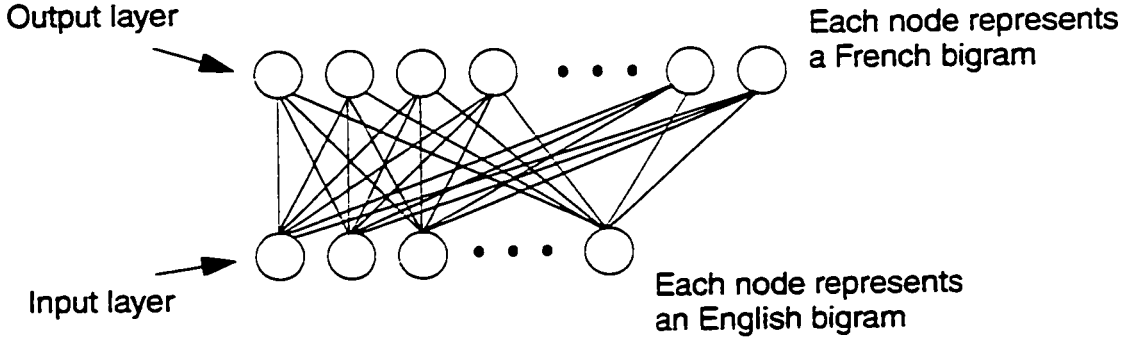
6.2.6 Building, training, and testing the neural network

Since we had placed the burden of representing word order on the data representation, our network architecture could be simple. As discussed in the previous section, it consisted of 465 input nodes (one for each bigram in the English text) and 506 output nodes (one for each bigram in the French text). We decided not to introduce any hidden layers.[9] This decision was consistent with the principle of economy for building neural networks which holds that one should keep the archi-

9. Had we been unable to train our network using this architecture we would have had the option of adding a hidden layer to increase the number of free parameters (i.e., connection weights) available to the network to learn the task.

tecture as simple as possible, and add complexity only when necessary. We also decided to use a fully-connected single layer feedforward network, and train it using the backpropagation algorithm.[10] The architecture of our resulting network (TransNet) is shown in Figure 13. It can be seen there that each element in the input layer is connected to each element in the output layer, and the network therefore by definition "fully connected".

Figure 13 TransNet architecture



The network shown in Figure 13 was built with a total of 5 commands using the Xerion backpropagation simulator. These commands are shown in Figure 14.

Figure 14 Xerion commands for building TransNet

```
uts_basicNet TransNet
uts_group TransNet.Input 465 {INPUT}
uts_group TransNet.Output 506 {OUTPUT}
uts_connectGroups TransNet.Input TransNet.Output
uts_connectGroups TransNet.Bias TransNet.Output
```

10. As it turned out, it was unnecessary to use backpropagation for training. Backpropagation was originally developed to overcome the problems of training multilayer neural networks and ours contain only a single layer. While backpropagation is effective at determining weights for single-layer networks, it is more resource-intensive than simpler algorithms such as the perceptron learning algorithm which is designed for use with single-layer networks.

In Figure 14, the first command "uts_basicNet" provides a name for the network. The next two lines beginning with "uts_group" instruct Xerion to create an input layer of 465 elements and an output layer of 506 elements. The "uts_connectGroups" commands connect each input element to each output element, and each output element to a "bias" element. The bias elements are created automatically by Xerion, and are required for training purposes. Van Camp (1995) provides a more detailed description of the Xerion network building command set. Training TransNet was done with the commands similar to the ones shown in Figure 15.

Figure 15 Commands to prepare TransNet for training

```
uts_randomizeNet TransNet
uts_exampleSet TransData
bp_netMinimizer mz
mz configure -net TransNet -exampleSet TransData
uts_loadExamples TransData -mchaffie/xertrn/trn00.bin
mz run
```

The first command in Figure 15 tells Xerion to set the initial link weights to random values. The next command "uts_exampleSet" declares a name for our training set. "bp_netMinimizer mz" generates a command called "mz" that will minimize the network error. The next line (beginning with "mz configure") configures the mz command to minimize the network error for TransNet using the data contained in the training set TransData. The command "uts_loadExamples" associates the training set name "TransData" with the contents of the file "trn00.bin", which contains our first training set. When the command "mz run" is executed, the training sentences in the file are automatically presented to the network one by one. Xerion calculates the overall network error and uses the backpropagation algorithm to decide how best to adjust the weights to reduce that error. Xerion continues to present sentences in the training set to the network and adjust weights

until the overall network error is reduced to close to zero, or cannot be reduced any more. Once the actual output of each output unit matches the unit's target output across all the sentences in the training set, the backpropagation algorithm is said to converge and the network is considered trained. If output value for one or more output units cannot be made to match the unit's target value across all the sentences in the training set, the algorithm does not converge. In this case, the network output may still be valuable as long as only a small number of output units cannot be properly trained. TransNet converged for 8 of our 10 training sets. In the two that did not converge, only 12 output units did not report the correct values for every sentence pair.

We trained the network 10 times, each time using a different training set. After each training run, we tested the network using the corresponding testing set. We wrote a tcl script to automate the process of testing the network (the script was called "pres_test.tcl"). Our script presented each input sentence in the test set to TransNet's input layer. For each sentence, TransNet activated a set of units in the output layer. The script recorded both the actual and target value for each element in the output layer across all the input sentences in the testing set; of course the network was not shown the target value—the script simply recorded this value for the purpose of comparison. The actual and target values were saved in a log file (named "tstres<n>", where <n> is the number of the testing set). A sample of the entries found in our log files is shown in Figure 16. Actual values are referred to as "output" values in the Figure.

Figure 16 Listing of target and actual values for some output layer elements

```
ExampleSet 37
Node: 0 target: 0.0, output: 0.0000703335
Node: 1 target: 1.0, output: 0.999853
Node: 2 target: 0.0, output: 0.0000504851
Node: 3 target: 0.0, output: 0.0000527501
```

The output of the network for each test sentence was thus a set of values for the elements in the output layer. If an output unit had a value greater than 0.5, its corresponding bigram was included in the network's translation of the input sentence. By converting all the units in the output layer which had a value greater than 0.5 into their corresponding bigrams, we were able to construct the set of bigrams that constituted the network output. An example of this is shown in Table 1.

Table 1: Sample of TransNet inputs and outputs

Input bigrams	Output bigrams
BG PARTIAL	BG DEGAGEMENT
CLEARING IN	EN SOIREE
IN THE	PARTIEL EN
PARTIAL CLEARING	SOIREE FN
THE EVENING	DEGAGEMENT PARTIEL
EVENING FN	

Note that the output bigrams shown are presented unordered. Once TransNet has chosen the appropriate bigrams, the task remains to reconstruct the sentence order. In many cases, we can reconstruct the sentence unambiguously simply by choosing bigrams that overlap (a bigram is said to overlap another if its second term is identical to the first term of another bigram). For instance, the only way to order the output bigrams in the Table above so that they overlap is: "BG DEGAGEMENT", "DEGAGEMENT PARTIEL", "PARTIEL EN", "EN SOIREE", "SOIREE FN". By removing the overlapping elements of each bigram, we arrive at the output sentence. The construction of output sentences from bigrams is treated in more detail in the "Discussion" section. The final step in testing the network is determining how well the network has translated the test sentences.

6.2.7 Measuring accuracy

With our testing output in hand, we were ready to determine the accuracy of TransNet. We needed to settle on a reasonable measure of error. We decided to apply the following criteria:

- An output unit with a target value of 1 and an output value greater than 0.5 was considered correctly selected.
- An output unit with a target value of 0 and an output value greater than 0.5 was considered incorrectly selected.
- An output unit with a target value of 1 and an output value less than 0.5 was considered incorrectly inhibited.[11]

We formalized these criteria as follows:

$$TA = \left(1 - \frac{\sum_{i=0}^{ts-1} \left(\frac{ni_i + ns_i}{nt_i} \right)}{ts} \right) \times 100$$

where

TA	=	TransNet Accuracy
i	=	testing sentence number
ni_i	=	number of bigrams incorrectly selected
ns_i	=	number of bigrams incorrectly inhibited
nt_i	=	number of bigrams in the target vector
ts	=	number of testing sentences

The expression within the inner brackets of our accuracy equation represents the error for a single output bigram set.

11. Output units with a target value of 0 and an output value less than 0.5 (that is, correctly inhibited units) were not included in our calculation. Because of the sparseness of our representation, most of the target values in a given testing set were equal to 0. If one included correct inhibition in the accuracy calculation, one could achieve what would look like a reasonably high rate of accuracy simply by setting the actual value of all the output units to 0.

We wrote an awk script (called "test_eval.awk") to automate the evaluation process. This script applied the accuracy formula described above to the testing results stored in the log files for each test set.

6.3 Results

Using the accuracy measurement presented in the previous section, our 10 testing sets had an average accuracy of 70.97%. The standard deviation was 3.23. The Table below shows the individual results for the training runs.

Table 2: Accuracy of 10 testing runs

Set Name	Accuracy
set 1 (evaluation set)	68.2212
set 2	72.8593
set 3	70.8163
set 4	71.6912
set 5	69.1339
set 6	73.3132
set 7	70.5090
set 8	72.8571
set 9	71.4505
set 10	68.8379

We calculated the standard deviation with the following formula:

$$s = \sqrt{\frac{\sum_{i=1}^{n} \langle x - \bar{\mu} \rangle^2}{n-1}}$$

where
s = standard deviation
μ = mean of data points
i = index
n = number of data points
x = a single data point

6.4 Observations

As previously mentioned, we decided to use the network weights generated by just one of the training sets to investigate what the network had learned. While ideally we would evaluate the network training results by examining the weights generated by all 10 training sets, we felt doing this would be impractical since each training set generated over 235,000 weights. We felt that the observations drawn from the examination of a single set of weights would generally extend to the other training and testing sets.

By examining the largest connection weights generated by our evaluation training set (the ones with a value greater than 5)[12] we uncovered three types of translational effects.[13] In decreasing order of frequency, but increasing order of interest, they are: a single bigram is translated by a fixed single trigram, a single bigram is translated by a fixed n-tuple of bigrams, and a single bigram is translated by one of several possible bigrams. The first two are within the power of the

12. As a crude point of comparison, the average of the positive weights on links between input and output units is 0.23. The average of the negative weights is -0.27. After examining the network output we determined that most of the interesting translational effects were realized through weights greater than 5. That is why we chose 5 as the cutoff for analysis.

13. There are other methods for investigating what the network has learned. One good method is to make progressively simpler networks (by preserving only links with large weights) and examine their relative performance. This way, the researcher can determine which links are critical, and from that develop a hypothesis about what the network is doing.

most primitive table-lookup translation system. In such a system, a word and its translation are permanently associated with one another. Every time a word appears in the source text the system looks up the translation in an equivalency table and places that word in the target text. This is the sort of dumb translation one might expect from a simple first-generation MT system. However, the third effect—a single bigram being translated by one of several possible bigrams—is far more interesting. It suggests that TransNet is capable of **selecting** target bigrams. MT researchers have been trying to build selectional capabilities into their systems for decades.[14]

6.4.1 Fixed one-to-one mappings

This first category is the most common and the least interesting. There are many examples in TransNet in which a particular bigram in the input layer is extremely heavily weighted towards a particular bigram in the output layer. The weighting is so great that the appearance of the input bigram at the input layer virtually guarantees the selection of the corresponding output bigram. We call this a fixed one-to-one mapping. The large number of fixed mappings is an artefact of the corpus we used. With its controlled vocabulary, it is written so as to be consistent in its terminology in both languages. The result is that a given lexical item in the source text is often translated by a fixed lexical item in the target text. This effect is reflected in the link weights of our network. For instance, the bigram "SHOWERS FN" has a link weight of 13.64 with the bigram "AVERSES FN". While it is possible that inhibitive connections (ones with negative weights) between "AVERSES FN" and some other bigrams in the input layer could prevent "AVERSES FN" from being selected even when "SHOWERS FN" appears in the input layer, it is not very likely: the

14. A typical rule-governed method for lexeme selection is to encode in the dictionary the *valencies* of a lexical item. For instance, the syntactic or semantic categories of the subject and object of a verb can be used to choose between its possible translations.

strongest inhibitive link to "AVERSES FN" is -7.203 (this is the bias weight). The next highest inhibitive weight is -1.72, which appears on the link to the input bigram "RAIN FN". In our evaluation testing set, "AVERSES FN" was correctly selected in 28 output sentences when "SHOWERS FN" appeared in the input sentence. However, even in our chosen sublanguage, one-to-one mappings are not absolute. For instance, the English fragment "...PROBABILITY OF FLURRIES OR SHOWERS FN" is translated by METEO as "PROBABILITÉ DE AVERSES DE NEIGE OU DE PLUIE FN". In this case, although "SHOWERS FN" does appear in the source text, "AVERSES FN" does not appear in the target text. TransNet was not able to make the appropriate selection in this case. In fact, what TransNet produced was "PROBABILITÉ DE AVERSES FN", losing "DE NEIGE OU PLUIE". This effect is likely due to the absence of the phrase "DE NEIGE OU DE PLUIE" in the training set. Given a larger training set, TransNet might well learn to account for this level of variation in translation.[15] Other examples of one-to-one mappings between bigrams in TransNet are the English bigram "SUNNY FN", which is strongly connected to the French bigram "ENSOLEILEE FN" with a weight of 10.72, and "DAYLABEL CLOUDY", which is connected to "DAYLABEL NUAGEUX" with a weight of 10.42.[16]

We noted that our representation scheme provided for some selectional capacity at the word, rather than bigram level. For instance, in our corpus the word "EVENING" is sometimes translated as "SOIR" and sometimes as "SOIREE". As it happens, the word "SOIR" always appears following the deictic "CE" to form the bigram "CE SOIR". The word "SOIREE" almost always

15. As it stands, this is still an interesting example of graceful degradation: even though the network had never before seen the input, it came up with a translation that might be better than nothing—which is what a rigid table lookup system might produce.

16. The bigram SUNNY FN unfortunately does not appear in the testing set (although it does appear in the training set). The bigram DAYLABEL NUAGEUX is correctly selected at the output layer for all 8 of the input sentences in the testing set that contained the bigram DAYLABEL CLOUDY.

appears following the preposition "EN" to form the bigram "EN SOIREE". The bigram "EN SOIREE" was correctly selected in 7 of 8 output sentences for which "(IN) THE EVENING" was present in the input sentence,[17] and the bigram "CE SOIR" was correctly selected in all 44 output sentences for which "THIS EVENING" was present in the input sentence. A simple word-for-word lookup scheme could not demonstrate this degree of flexibility, since "EVENING" would be have to be permanently associated with either "SOIREE" or "SOIR". The network thus exhibits some ability to cross word boundaries in determining the translation for a word. Again, this is a result of our bigram representation scheme, which itself crosses word boundaries, rather than the network. One might expect that a trigram representation would display the same kind of behaviour across bigram boundaries; however, an investigation of this possibility is beyond the scope of our thesis.

6.4.2 Fixed one to n-tuple mappings

We also noted that TransNet had the ability to map a single input bigram to an n-tuple of several output bigrams. The best example of this is perhaps the mapping of the English bigram "SOME RAIN" to the French bigram set "UN PEU", "PEU DE", "DE PLUIE". The connection weights between "SOME RAIN" and these three French bigrams are, respectively, 9.91, 10.32, and 9.19. They are by far the three highest weights for the English bigram (the next highest is 5.37 for "NUIT UN"). These weights suggest that TransNet has strongly associated the three French bigrams with their English equivalent. Of the 3 times that "SOME RAIN" appears in the testing set, it is correctly translated twice. In the instance in which it is incorrectly translated, the bigram "DE PLUIE" is incorrectly inhibited while the other two bigrams "UN PEU" and "PEU DE" are

17. The sentence for which EN SOIREE was not correctly selected also contained the bigram CE SOIR. EN SOIREE was not correctly selected in part because they are mutually exclusive in the training set.

correctly selected. Though a one-to-several mapping could be built into a table lookup scheme using a dictionary that lists phrases, the network seems to be discovering the phrases for itself. This is an interesting point, since the task of compiling a table of source language words and phrases and their target language phrasal equivalents is far from trivial.

6.4.3 Flexible one-to-one mappings

Flexible one-to-one mappings occur when a single input bigram can be translated by different single output bigrams. The decision of which bigram to include in a given translation cannot be made without information about the other bigrams that are present in the sentence; that is, the decision is conditioned by context. The best example of this is the collection of input bigrams that end with the pronoun "THIS". These bigrams include "EARLY THIS", "LATE THIS", "CLEARING THIS", and "SHOWERS THIS". They typically occur in phrases such as "EARLY THIS AFTERNOON", "EARLY THIS MORNING", "SHOWERS THIS EVENING". The interesting aspect of these bigrams from a translation standpoint is that the bigram "EARLY THIS" can be translated by several alternative French bigrams. The two most common are "TOT CE" and "TOT CET". The proper choice depends on the noun that appears next in the sentence (i.e., the noun in the bigram that immediately follows "TOT CE" or "TOT CET"). If it is masculine and begins with a vowel (like "APRES-MIDI), then "TOT CET" is the appropriate translation. If it is masculine and begins with a consonant (for example, "MATIN"), then "TOT CE" is the correct choice. For our evaluation set, "TOT CET" is correctly selected 5 out of 6 times. "TOT CE" is correctly selected 13 out of 16 times. Unfortunately, in this case the connection weights do not tell a simple story. Both "TOT CE" and "TOT CET" are strongly associated with "EARLY THIS" (with weights of 9.34 and 7.90 respectively). This is what one might expect since "EARLY THIS" is very likely to

select one of the two bigrams in the output layer. However, the criteria the network is using to choose between the two is not evident in the connection weights. One might expect the bigram "THIS MORNING" to be strongly associated with "TOT CE", since "TOT CE" only appears in the training set alongside "CE MATIN". However this is not the case. In fact the strongest weighting for "TOT CE" appears on its link with the input bigram "KM/H EARLY". As for "TOT CET", there do not seem to be any predominant weights on its connections with the input bigrams. So, TransNet appears to solve the problem of selection for the bigram "EARLY THIS" but we do not know for certain what criteria it is using to make the selection. In the case of the bigram "TOT CE", it may well be that TransNet has learned to select the bigram "TOT CE" if the bigram "KM/H EARLY" co-occurs with it. If this is the case, this outcome would simply be an artefact of our particular training set: the bigram "KM/H EARLY" happened to appear in sentence pairs where "TOT CE" also appeared, but happened to be absent in sentences where "TOT CET" appeared. Of course this is only a hypothesis; the correspondence inferred by TransNet using our evaluation training set may be much more complicated.

Since we do not know how the network is making its selection, we cannot be sure that it would reliably translate the bigram "EARLY THIS" in every possible context. More research must be done to determine the criteria used by TransNet in making its decisions. Nonetheless, it is clear that our network is capable of selecting target bigrams based on the presence of other bigrams in the sentence. That is, it demonstrates the ability to make context-dependent decisions.

In examining the networks, we also noticed some unexpected relationships. For instance, the bigram "BY NUM" was most strongly associated with the French bigram "GRADUEL CE" (with a link weight of 13.26). It would be more reasonable to expect the bigram "BY NUM" to be

strongly associated with "DE NUM" since the latter is the most common translation of the former, but the link between these bigrams carried a much smaller weight (5.72). This suggests that the network may be learning associations that are not valid. This problem is likely compounded by the small size of the training set used for TransNet. For instance, "GRADUEL CE" appears only once in the training set, and the sentence in which it appears in also contains the bigram "DE NUM". Training the network with a larger data set could very well remove these inconsistencies. This is something well worth testing.

We also noticed associations that suggested the network had expectations about which bigrams were likely to appear in the output layer given a particular single bigram in the input layer. For instance, the bigram "BG TODAY" was strongly associated with both "BG AUJOURD'HUI" and "AUJOURD'HUI PLUIE" (with link weights of 12.80 and 5.04 respectively). From this we can infer that the network exhibits some associative behaviour. For instance, there is a strong association between the bigram "SUNNY WITH" and the bigram "ENSOLEILLE AVEC" (the link weight is 5.77), as one might expect. However, it is also strongly associated with the output bigram "AVEC PASSAGES" (link weight 5.76), and somewhat less so with "PASSAGES NUAGEUX" (link weight 3.35). When the network is presented with the input bigram "SUNNY WITH", it seems to strongly expect at least two bigrams in the output layer: "ENSOLEILLE AVEC" and "AVEC PASSAGES". It expects "PASSAGES NUAGEUX" somewhat less strongly, but still more strongly than any other output bigram. A glance at the corpus shows that when "SUNNY WITH" appears in the input sentence, "ENSOLEILLE AVEC PASSAGES NUAGEUX" (a collection of the three bigrams mentioned above) almost always appears in the output. Just presenting the one bigram, seems to prepare the network to see several others. With a

more varied training set, one could make an interesting study of whether a strong association between a single input bigram and several output bigrams corresponds to collocations and fixed expressions in a sentence. Finding collocations and phraseology is an important task for lexicographers, and there is work going on to automate it (Langlois 1997).

There is also the possibility that the weighting between certain bigrams was influenced by the time of year at which the training corpus was compiled and the region from which its composite weather forecasts were collected. This makes sense: for example, if the training corpus is composed of forecasts collected during a particularly rainy summer in a rainy region, it is likely to contain many occurrences of the word "RAIN" in a variety of bigrams. A network trained on this corpus would undoubtedly learn a great deal about the relationships between the different bigrams containing "RAIN" or "PLUIE". A network trained on a corpus collected in a hot sunny area would presumably learn more about the relationships between bigrams that contained "SUNNY" or "ENSOLEILLE". In our corpus the bigram "BG TODAY" is strongly associated with both "BG AUJOURD'HUI" (link weight 12.80) and "AUJOURD'HUI PLUIE" (link weight 5.72). It is much less strongly associated with "AUJOURD'HUI ENSOLEILLE" (link weight 2.72). This suggests that the network has learned that "today" is more often found in sentences with the word "rainy" than in sentences with the word "sunny". It is important to remove this type of effect from the corpus, and the best way to do so would be to build the corpus using weather forecasts from a random sampling of regions over the space of a year or more. Nonetheless, it is important to remember that although the network seems to lean towards the bigram set "BG AUJOURD'HUI" and "AUJOURD'HUI PLUIE" given the presence of "BG TODAY" in the input set, it is flexible enough to render the bigram set "BG TODAY" "TODAY SUNNY" as "BG AUJOURD'HUI

ENSOLEILLE" whenever it needs to (this construction occurs twice in the evaluation testing set and is correctly selected in both cases). This is a result of the very strong association between "TODAY SUNNY" and "AUJOURD'HUI ENSOLEILLE" (link weight 10.63).

6.5 Discussion

There are several important areas of discussion. They are grouped into six sections: reconstructing sentences, corpus variability, sample size, scalability, conceptual mapping, traditional concerns, and research emphasis.

6.5.1 Reconstructing sentences

Reconstructing sentences from bigrams can be a complicated process. Fifty-four of our output sentences (27%) could not be reconstructed unambiguously from the bigrams selected by the network. The main problem was the data representation we chose. Each bigram is represented by a single output element. If a bigram appears more than once in a sentence translation, it is represented by the activation of only a single element in the output layer. When an element has been activated in the output layer, one knows only that the corresponding bigram appears at least once in the sentence; one does not know with certainty how many times it ought to appear. Consequently, reconstructing a sentence becomes difficult if the bigram ought to appear several times. For example, the sentence "VENTS DE NUM A NUM KM/H DIMINUANT A NUM A NUM EN MI-JOURNEE." contains three instances of the bigram "A NUM", and two instances of the bigram "NUM A". This sentence cannot be reconstructed easily from its composite bigrams. We have developed guidelines which predict whether a sentence can be reconstructed from its composite bigrams. These guidelines are tied to three simple rules we deduced while attempting to reconstruct sentences. These rules are:

- **Rule 1** (the "full use" rule): One must use all the bigrams activated by the network.
- **Rule 2** (the "no loopback" rule): When given a choice between using a bigram that has already been used in the sentence and one that has not, one must choose the one that has not.
- **Rule 3** (the "re-use rule"): One may reuse a bigram in the sentence if no unused bigrams can be used in that position.

With these rules in mind, we have developed the guidelines shown in Table 3 for sentence reconstruction.

Table 3: Guidelines for sentence reconstruction using bigrams and their component words

Condition	Difficulty	Rules
one word appears twice	low	1 and 2
bigram appears twice	low	1, 2, and 3
bigram appears more than twice	high	unknown
more than one word appears twice	low	1, 2, and 3

There are several options that suggest themselves to help future researchers surmount the problem of sentence reconstruction. If the researchers decide that allowing for several tokens of a word type in a sentence is important, they can represent the sentences by trigrams instead of bigrams. Trigrams allow for greater repetition within the sentence while maintaining better reconstructability. For example, all but 8 of our 200 output sentences (4%) could have been reconstructed had we used a trigram representation. The guidelines for sentence reconstruction using trigrams are shown in Table 4.

Table 4: Guidelines for sentence reconstruction using trigrams and their component words

Condition	Difficulty	Rules
one word appears more than once	low	no special rules
bigram appears twice	low	1 and 2

Table 4: Guidelines for sentence reconstruction using trigrams and their component words

Condition	Difficulty	Rules
trigram appears twice	low	1, 2, and 3
trigram appears more than twice	high	unknown
more than one bigram appears twice	low	1, 2, and 3

Although we cannot prove it conclusively, these rules appear to generalize as more words are added to the unit of translation. For instance, if one were to use word quintuplets as the unit of translation, then sentences could be reconstructed even if several trigrams appeared more than once. However, a word of caution: as the size of the unit of translation is increased, the dictionary size is likely to increase as well. For instance, in our corpus there are only 672 English bigrams but 1272 English trigrams. The network size will increase geometrically, necessitating a larger corpus and more processing power, disk storage, and memory. Therefore an optimal size of translation unit must be sought—probably trigrams.

Another option that presents itself to future researchers is to modify the corpus to reduce the number of words that are repeated. For instance in our corpus, the word "de" is often repeated several times in a sentence. In most cases it adds little to the meaning of the sentence and could easily be ignored. A simple way to do this would be to replace the word "de" and whatever word follows it with a unified label. For instance, in the sentence fragment "NUM POURCENT DE PROBABILITE", one could replace "DE PROBABILITE" with the label "PROB". So the sentence would become "NUM POURCENT PROB". When reconstructing the sentence, one could simple substitute "DE PROBABILITE" when a bigram that contained "PROB" was activated. For instance, in the above example, "POURCENT PROB" would be a bigram. If that bigram were selected, one would substitute the words "POURCENT DE PROBABILITE". Since connection words like "de"

are likely to appear several times in many sentences, this method is likely to reduce significantly the number of ambiguous sentence reconstructions; however, it is unclear what effect this type of data manipulation would have on network learning.

The final option for researchers who opt for the bigram representation is to develop more sophisticated rules for reconstructing sentences. The three rules we constructed are rather simple. They were effective for our corpus, but they may not be sufficient for other corpora.[18] By further investigating their corpora, researchers might find more sophisticated or more appropriate rules for determining how bigrams should be put together to build a sentence. Perhaps another neural network could be used to reconstruct the sentences. Furthermore, researchers who do not want to incorporate word order into their data representation might incorporate it into the network structure instead, and attempt to solve the problem that way.

6.5.2 Corpus variability

The variability of the corpus was quite low. For instance, in our corpus there was a sentence-by-sentence correspondence between source and target sentences. That is, each sentence in the source text was translated by a single sentence in the target text. This rigid relationship between source language (SL) and target language (TL) sentences is not common in general language. In addition, many of the factors that complicate other MT systems were not present in the sublanguage of our corpus. While we did not ourselves do an in-depth lexical analysis of the corpus, it is known that its sublanguage contains little or no lexical ambiguity, including syntactic category ambiguity, homography, or polysemy. Structurally ambiguous sentences are also difficult if not impossible to

18. Our use of placeholders for some words (such as the days of the week) aggravated the reconstruction problem by adding ambiguity to the sentence. Another way to simplify the reconstruction would be to avoid the use of placeholders.

find (Lehrberger 1980, 99f.). There are no long noun phrases and no anaphoric references. In short the corpus text was extremely consistent. This is not characteristic of natural language in general. In fact, inconsistency, anaphora, etc., are the very characteristics that defeat many MT systems.

Hence, we think of our network only as a proof-of-concept. Having lent some credence to the notion that neural networks can successfully be applied to natural language translation using our representation model, we feel it would be a very interesting next step to test our approach against a corpus of more variable natural language. However, given the current constraints on network size, we feel that representing free-form natural language translation is still beyond the reach of our neural network model. The application of neural networks to natural language should therefore be restricted to sublanguages, at least until the construction of larger-scale neural networks becomes feasible.

Another characteristic inherent in our choice of sublanguage is the small vocabulary size (at 143 English words and 149 French words). We consider this to be a strength of our corpus selection. A restricted vocabulary is not invalid if the vocabulary is restricted naturally. The vocabulary size in our experiment was restricted naturally as a function of the meteorological sublanguage contained in the METEO weather reports. People do not expect or need a large vocabulary to communicate the weather to one another. As a result, the output produced by TransNet was not constrained in any way that made it less understandable to human beings than any other METEO weather report.

A related point is that the translation METEO uses for a given sentence is very consistent. That is, in our corpus a sentence is always translated the same way. In no cases did there exist two differ-

ent translations for the same source sentence. It is reasonable to ask how a neural network would fare given a corpus of freer translations. However, it is also fair to point out that lack of variation is relatively common in real-world translation, particularly in technical fields. Given a sentence in a particular context, an experienced translator often translates it the same way. In fact, lack of variation is part of what constitutes a translator's *style*. For researchers planning to apply the neural network approach to more widely varying text, we recommend maintaining a high level of translation consistency in their corpus. There are two simple approaches one can take: one might use source and target text produced by a very small number of translators, or one might use source and target text from a large group of translators who all follow the same strict guidelines for translating text.

This leads us to another point—it may seem strange that we have used a corpus of computer-generated translations to train a neural network which is intended to generate human-like translations. However, it is important to remember that the output of METEO is the result of thorough analysis of the translations of weather reports done by human beings. By most accounts the output of METEO is indistinguishable from that previously produced by human translators (Chandioux 1976). As a result, we feel that using METEO as the source of our bitext corpus is virtually equivalent to using the output of human translators. A follow-up question might be, "What is the value of TransNet if all it can do is perform a subset of the translation that METEO already performs?" The answer is that TransNet was created to test the hypothesis that neural networks could be applied to natural language translation, not to compete with an existing software product. METEO proved that a procedural computer program was capable of translating a natural sublanguage successfully. The TransNet experiment was designed to find out if a neural network could

also be used for this task. The fact that TransNet was able to translate the majority of the sentences presented to it does not mean that the neural network approach is better than, or even equal to, procedural programming; just that it is possible.

6.5.3 Sample size

As it turned out, we would have benefited from a larger sample size. Although we collected weather reports for a full month several times a day, we ended up with a sample size that was too small to fully evaluate the performance of our neural network. TransNet consists of over 230,000 links. Some sources even recommend using a training set that consists of 30 times as many training pairs as there are links in the network to eliminate the possibility of overfitting the data[19] (Sarle 1997). However, the number of links in the network was determined in large part by our data representation. Since we did not yet know what our data representation would look like when we collected the data, we did not know how much to collect. We also had time constraints and disk space limitations. Also, while we originally collected approximately 10,000 sentences, it turned out that only about 10% were unique (which is not surprising given our chosen domain of weather reports). This is one instance where the lack of variability in the METEO output worked against us. There are broader implications too for the applicability of neural networks to natural language translation. If such a large amount of data is necessary to train our modest network, then the amount of data needed to train a network to translate text from a richer domain such as medicine or law might be prohibitive (assuming that a suitable bitext corpus could be constructed in the first place). The problem of finding adequate training information is typical of the neural net-

19. When data is overfitted, the network tends to generalize based not only on valid training examples, but also on spurious ones.

work approach in general. Neural networks learn by example. Therefore the approach should not be applied to fields in which one cannot find a sufficient number of examples.

The size of our network (that is the number of links it contains) was a reflection of the data representation we chose. Perhaps other researchers could reduce the required size of the training and testing corpus by finding a more compact representation. A more compact representation would result in fewer network connections and thus more modest data requirements. Regardless, we recommend that researchers collect approximately 10 times more data than they think they will need. They must keep in mind, however, that this will give rise to the problems implicit in manipulating and managing large corpora. Disk space is only the beginning. CPU speed, RAM, and the maximum file size supported by the principal software applications are also important considerations.

When evaluating the performance of a neural network, it is common to train the network on a third of the example set, and test it on two thirds. For each of our 10 training and testing sets, we split the bigram corpus into two equal-sized sets, one for training and one for testing. We felt that this small break with tradition was not significant, given that the number of sentence pairs in each set was already very small relative to the number of links in the network.

6.5.4 Scalability

A related point is the scalability of our approach. After all, if we were not able to build a 700 x 700 simple feedforward network due to computing constraints, how could we hope to build a network able to translate sentences from a linguistically rich domain? For one thing, while our computing resources were the equal of those used by many professional programmers, they were not the most powerful available. A fully funded research project would likely be able to afford more

intimidating horsepower. Also, neural network processing is inherently parallel. We had to use a serial computer to approximate a parallel one. Serial computers are not designed to perform parallel computations efficiently. A neural network processor capable of handling multiple simultaneous operations might have allowed us to use a larger network. Another, perhaps simpler, way of reducing the need for computing resources is to build a neural network simulator that is optimized to train and run TransNet. Xerion is a general purpose neural network simulation environment; it was designed to be useful to neural network researchers in a large number of fields with a wide variety of requirements. Software specifically written to implement TransNet could take advantage of the particularities of our network architecture and data representation. This could considerably reduce the resource requirements of the network.[20]

On the question of scalability, one small advantage of the bigram approach is that it is theoretically bounded for any given domain. For instance, if a sublanguage contains 1,000 words, the largest number of bigrams it can contain is 1,000**2 or 1 million. The number of bigrams that are actually used is much less than that (for example, the sentence fragment "FOGGY A" does not appear as a bigram anywhere in our corpus, although it is theoretically possible). While it is currently impossible to build a network of even 500,000 input elements, networks of this size may come within reach as processing power continues to increase.

6.5.5 Conceptual mapping

It should also be noted that given our method of representing data, the link between related words is not apparent. For instance, the network has no way of knowing that "CETTE NUIT" and "LA NUIT" have the word "NUIT" in common. We do not find this to be of particular concern. The

20. In fact, Dr. Mario Marchand is currently pursuing this approach for TransNet.

network neither knows nor needs to know. It was not designed to reflect the intricacies of human language processing. Indeed, it would not be surprising if a neural network used a technique alien to human beings to store and process natural language; we have remarked earlier that connectionist researchers are sceptical nowadays about the notion that neural networks mimic human thought processes.

6.5.6 General concerns

As mentioned in the Introduction, there are some well-known disadvantages to neural networks. For one thing, it is sometimes very difficult to predict without actually performing the computations what output the network will produce given unseen input. This may make this approach inappropriate for applications in which inaccurate output could cause a catastrophe. For instance, one would not want a neural network in charge of the reactor core in a nuclear power station, nor in charge of translating the reactor's operating manual. One should remember though that it can also be difficult to predict what decisions will be made by complex rule-based algorithms.

A neural network lacks the transparency of an algorithmic system. On the other hand, it is not always necessary to understand a process as a sequence of steps in order to benefit from that process. For example, people often ride bicycles as a means of locomotion, but most would be hard-pressed to describe the steps involved. Despite the fact that a neural network does not make explicit the steps involved in a process, it can still be used to perform useful tasks.

6.5.7 Research emphasis

Part of the motivation for this thesis was the difficulty of developing sufficiently flexible rules for translating natural language sentences. We thought it might be easier to have the computer do the

work of analyzing text and then inducing rules for reproducing the text in another language. However, as it turns out this approach does not make research any easier: the emphasis shifts from linguistic analysis to building appropriate corpora, conditioning corpus text, developing data representations, designing network architectures, and building, training, and testing networks. The amount of effort expended in these activities is easily equivalent to that spent analyzing natural language to extract rules and then implementing the rules in a programming language. In short, the neural network approach is sadly not the lazy man's substitute for morphological, syntactic, and semantic textual analysis. On the other hand, the onerous steps involved in implementing the neural network approach are mechanistic and automatable. Rule extraction is an art, and thus inexact, error-prone, and incomplete.

CHAPTER 7: RECOMMENDATIONS

Since network size is limited by available computing resources, researchers should expect to have to limit the vocabulary contained in their bitext corpora. One way to do this is to choose an appropriate sublanguage so that the vocabulary is restricted naturally. Another way is to ensure that the translations included in their bitext corpora are written in an artificially controlled language.

Researchers should also collect much more data than they think they will need. The rule of thumb in neural network research is that the training set size should be about a third as large as the number of links in the network. When presented with a small training set, a network sometimes *overtrains*. An overtrained network infers invalid rules from the training set based on a very few erroneous data points. The most telling outcome of an overtrained network is lack of generalization. In our case, the network did not seem to exhibit this behaviour. We had reasonably good generalization (the network translated sentences that it had not encountered before with an average accuracy of 70.97%). The number of links in the network is a function of data representation and network architecture, and these will undoubtedly change over the course of any research (they certainly did for us). As a result, it is difficult to predict just how large a corpus one will have to collect. It is best to err on the side of excess.

It would also be wise to investigate the corpus thoroughly before choosing a data representation and network architecture. In our case, the choice of bigrams as the unit of representation meant that some sentences could not be reconstructed from the network output. A bigram representation would be appropriate if the corpus consisted of extremely simple sentences with very little repeti-

tion (see Section 6.5 for more information). For more complex sentences, perhaps a trigram representation (or even a quadrigram representation) would be more appropriate.

Computing resources were crucial in our experiment. While much more powerful than most home computing systems, the computers we used were not the most advanced available. We strongly recommend that researchers plan to invest significant funds in computing resources. Faster computers will significantly reduce the amount of time spent waiting for networks to train. Our training runs with Xerion often lasted more than 24 hours.

7.1 Improving accuracy

The ultimate goal of further research into our approach should be improved translational accuracy. We feel there are four main ways to improve our results:

- **Use more sophisticated validation methods**—Each training set for TransNet consisted of half the corpus sentence pairs. The rest of the corpus sentence pairs were placed in the corresponding testing set. Because of the small size of our corpus, bigrams that appeared in the training set were sometimes absent in the testing set, and, more important, bigrams that did not appear in the training set sometimes appeared in the testing set. Having never been exposed to them during during training, TransNet was not able to correctly translate these bigrams in the testing run.[21] A simple way to improve results, therefore, is to increase the size of the training set relative to the initial testing set. Perhaps the best way to do this is to use leave-one-out cross-validation. In this validation scheme there are as many training and testing sets as there are sentence pairs in the corpus. Each training set contains all but one of the sentence pairs in the corpus. The single remaining sentence is the testing set for that training set, and all the testing sets are disjoint (that is, they each contain a different sentence pair). The

21. For example, there were a total of 315 English bigrams in our evaluation training set. The entire corpus contained 465 English bigrams. This means that in the testing session, TransNet encountered 150 bigrams that had not been presented to it in the training session.

accuracy of the network is the average of the accuracy for each training and testing set. This method maximizes the number of bigrams in the training set, which makes it suitable for use with small corpora.

- **Increase the size of the corpus**—Increasing the size of the corpus would lead to larger training sets. If presented with larger training sets, TransNet would encounter individual bigrams in a greater variety of sentences, which would likely increase its capacity for generalization.
- **Increase the size of the translation unit**—As discussed in Section 6.5.1, increasing the size of the translation unit increases the translational capacity of the network and simplifies sentence reconstruction.
- **Use more sophisticated training methods**—We used the backpropagation learning algorithm largely because of its accessibility. There are, however, learning algorithms that seem more appropriate for handling the special challenges of connectionist MT. One of the characteristics that separates connectionist translation systems from other types of connectionist systems is that the value of each output unit in a translation system is governed by only a few input units. In extreme cases, as discussed in Section 6.4.1, an output unit may only ever be activated if one particular unit is present at the input layer (the fixed one-to-one correspondences). In other connectionist disciplines, researchers strive to ensure that individual connections between units do not carry overwhelming significance for any single output unit. For connectionist MT, then, it is worthwhile to investigate the use of other algorithms, such as the Winnow algorithms developed by Nick Littlestone (Littlestone 1988), which can be proven mathematically to perform well when the number of relevant inputs is small.

CHAPTER 8: CONCLUSION

We have shown that neural networks can be applied to the task of natural language translation, at least to a limited extent. However, obstacles still present themselves to the researcher who would implement a full-scale natural language translation system. The greatest of these is perhaps the limitation on network size that results from the lack of available computing power. Natural languages typically have core vocabularies of several thousand words. It is hard to imagine a network of even 1,000,000 links successfully translating free-form text from one language to another given the huge number of vocabulary elements involved. Reined in by this computational constraint, researchers will likely be limited to applying neural networks to natural language translation on a relatively small scale. One good way to reduce the scale of the application is to have the network perform its translation in a sublanguage environment. However, sublanguages that reduce vocabulary and lexical and semantic ambiguity to the necessary degree may be hard to find. And once they are found, researchers must somehow build large corpora of aligned bilingual text taken from these domains. This task has become easier in recent years with the introduction of software that performs the alignment automatically, but researchers will still need to root out a large enough sample of electronic text in two languages.

Still, increasing the scale somewhat should be possible. When this thesis was begun three years ago, research systems with 100 MHz processors and 32 MB of RAM were rare. They are still rare now, but for a different reason—they are outdated. A Pentium-class computer with a 200 MHz processor and 128 MB of RAM is not uncommon, and can be had for a reasonable price. With these more powerful computers and application-specific software, researchers should be able to build networks many times the size of TransNet without much trouble. This would allow them to

evaluate the bigram or trigram approach in the face of much more varied bitext. Challenging our approach with a bigger vocabulary and greater ambiguity is clearly the next step. Success there would bode very well.

But there is not only our approach to investigate. Our literature review makes it clear that this field is still in its infancy. This thesis is, we believe, the first to suggest a bigram or trigram data representation, but our time and resources were very limited. What interesting results might be obtained were a well-funded professional researcher to apply his or her talents and resources to the approach we have presented here?

REFERENCES

Allen, Robert B. (1987). "Several Studies on Natural Language and Back-Propagation." First International Conference on Neural Networks. San Diego, 335-341.

Brown, Peter F. *et al* (1993). "The Mathematics of Statistical Machine Translation: Parameter Estimation." Computational Linguistics.19:2.263-301.

Brown, Peter F. *et al* (1994). "Automatic Speech Recognition in Machine Aided Translation." Computers, Speech and Language. 8:3.177-187.

Carling, Alison. "Introducing Neural Networks." Wilmslow: Sigma Press. 1992.

Chalmers, David J (1990). "Syntactic Transformations on Distributed Representations." Connection Science. 2:1 & 2.53-61.

Chandioux, John (1976). "METEO: un système opérationel pour la traduction automatique des bulletins météorologiques destinés au grand public." Meta. 21:2.127-133.

Chrisman, Lonnie (1991). "Learning Recursive Distributed Representations for Holistic Computation." Connection Science. 3:4.345-366.

Elman, Jeffrey L. (1990a). "Finding Structure in Time." Cognitive Science 14:179-211.

Elman, Jeffrey (1990b). "Representation and Structure in Connectionist Models." In G.T. Altmann (ed.) Cognitive Models of Speech Processing. Massachusetts: MIT Press.

Fodor, Jerry A. and Pylyshyn, Zenon W. (1988) "Connectionism and cognitive architecture: A critical analysis." Cognition 28:3-71.

Gallant, Stephen I. (1991). "A Practical Approach for Representing Context and for Performing Word Sense Disambiguation Using Neural Networks." Neural Computation 3:293-309.

Gallant, Stephen I. (1993). "Neural Network Learning and Expert Systems." Cambridge: MIT Press.

Hartmann, R. R. K. and F.C. Stork (1972). "Dictionary of Language and Linguistics." London: Applied Science Publishers.

Harris, Brian (1988, March). "Bi-text, a new concept in translation theory." Language Monthly 54.8-2.

Hecht-Nielsen, Robert (1990). "Neurocomputing." Don Mills: Addison-Wesley.

Hutchins, W.J. (1986). "Machine Translation: past, present and future". Chichester: Ellis Horwood

/ New York: Halsted.

Jordan, M. I. (1986). "Serial order: a parallel distributed processing approach (ICS Technical Report 8604)." Institute for Cognitive Science, University of California, La Jolla, California.

Kimura, Kazuhiro and Suzuoka Takashi (1992). "Association-based Natural Language Processing with Neural Networks." Proceedings of the 30th Conference of the ACL. Newark, Delaware. 224-231.

Kittredge, R. and J. Lehrberger (eds.) (1982). "Sublanguage." Berlin: De Gruyter.

Langlois, Lucie (1997). "Bitexte, bi-concordance et collocation". Unpublished MA dissertation. School of Translation and Interpretation, University of Ottawa.

Lehrberger, John (1982). "Automatic Translation and The Concept of Sublanguage". In R. Kittredge and J. Lehrberger, eds., Sublanguage. Berlin: De Gruyter: 81-106.

Lingaard, R., Myers, D.J. and Nightingale, C. (1992) "Neural Networks for Vision, Speech and Natural Language." New York: Chapman & Hall.

Lisboa, P.G.J. (ed.) (1992). "Neural Networks: Current Applications." New York: Chapman & Hall.

Littlestone, Nick (1988). "Learning quickly when irrelevant attributes abound: a new linear threshold algorithm." Machine Learning. 2:285-318.

Marchand, Mario and Moustefa Golea (1993). "A Constructive Algorithm for Neural Decision Lists". Proceedings of the 1993 Annual Meeting of the International Neural Network Society. Hillsdale, NJ: Erlbaum Associtates. 3:560-563.

McLean, Ian J. (1992). "Example-Based machine Translation using Connectionist Matching." Proceedings of the 4th International Conference on Theoretical and Methodological Issues in Machine Translation. Montreal: 35-43.

Muller, Berndt and Reinhardt, Joachim (1990). "Neural networks: an introduction." New York: Springer.

Nelson, Marilyn McCord and Illingworth, W.T. (1990). "A Practical Guide to Neural Nets." Don Mills: Addison-Wesley.

Nida, Eugene A., and Charles R. Taber (1969). "The Theory and Practice of Translation". Leiden: Brill.

Pollack, J. (1988). "Recursive auto-associative memory: devising compostional distributed representations". Technical Report MCCS-88-124, Computing Research Laboratory, New Mexico State University, Las Cruces.

Rumelhart, David E., and McClelland, James L. (eds.) (1986). "Parallel Distributed Processing: Explorations in the Microstructure of Cognition." Cambridge: MIT Press.

Sadler, B. C. V. and A. P. M. Wikkam (1986). "Word Expert Semantics: an Interlingual Knowledge-Based Approach" (Distributed Language Translation 1). Dordrecht: Floris.

Sarle, Warren S. (1997, June). "Neural Network FAQ". Internet newsgroup comp.ai.neural-nets.

Sharkey, Noel E. and Ronan G. Reilly (eds.) (1993). "Connectionist Approaches to Natural Language Processing." Hillsdale: Lawrence Erlbaum Associates.

Van Camp, Drew (1995, March). "A Users Guide for the Xerion Neural Network Simulator Version 4.1". Department of Computer Science, University of Toronto.

Wasserman, Philip D. (1989). "Neural Computing: Theory and Practice." New York: Van Nostrand Reinhold.

Wermter, Stefan. (1994). "Learning Natural Language Filtering Under Noisy Conditions." Proceedings of the 10th IEEE Conference on AI for Applications. San Antonio, 215-221.

Winston, Patrick Henry (1977). "Artificial Intelligence." Don Mills: Addison-Wesley.

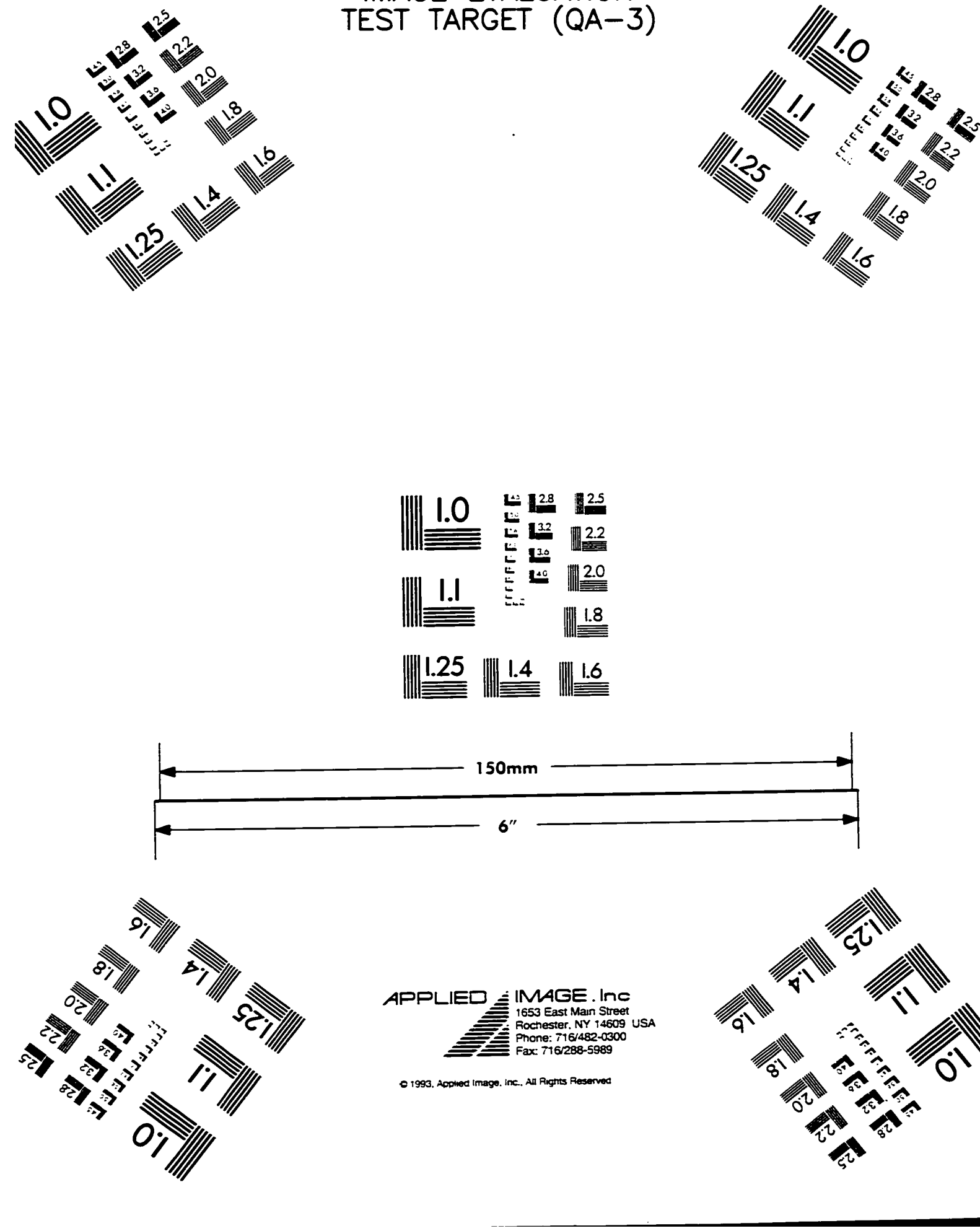


IMAGE EVALUATION
TEST TARGET (QA-3)
150mm
6"
APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989
© 1993, Applied Image, Inc., All Rights Reserved