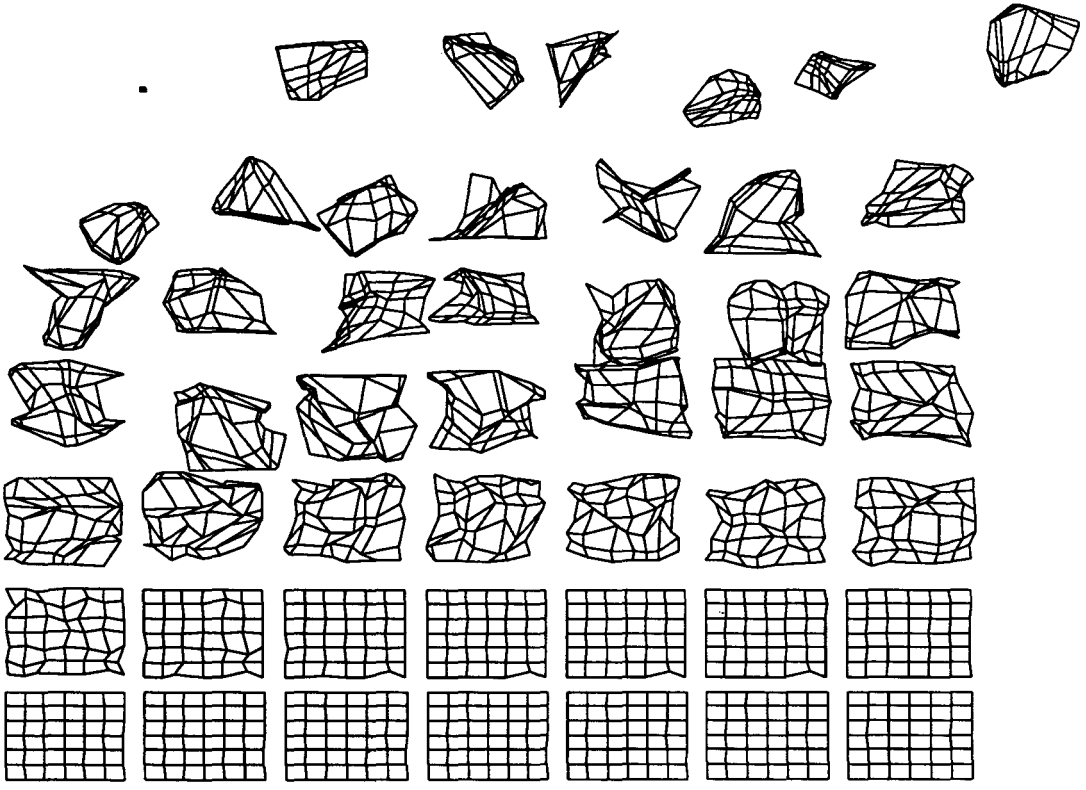
Neural Networks in Natural Language Processing and Information Retrieval



Johannes Cornelis Scholtes

Neural Networks in Natural Language Processing and Information Retrieval

Academisch Proefschrift
ter Verkrijging van de Graad van Doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus,
Prof. Dr. P.W.M. de Meijer,
in het openbaar te verdedigen
in de Aula der Universiteit,
(Oude Lutherse Kerk, ingang Singel 411, hoek Spui)
op donderdag 21 januari 1993 te 13.30 uur,

door Johannes Cornelis Scholtes
geboren te Leiden

Voor Frédérique

Table of Contents

Preface

In this thesis, applications of neural networks in natural language processing and information retrieval are discussed. These neural techniques have shown considerable success for many other unstructured data sets. However, most of those applications were involved in low-level signal processing such as robot-arm movements or image processing. Here, it is argued that neural techniques can also be used for particular tasks in higher level cognition such as language processing.

In natural language processing, a computer system analyses language and demonstrates understanding by invoking the proper actions. In information retrieval, a large unstructured collection of natural language data is made accessible by a computer system, which must provide the user with relevant text passages according to some query. The main problems in both applications are caused by the fact that natural language displays ambiguities on all levels, leading to an enormous number of possible interpretations in the processing as well as in the retrieval of it. In one way or an other, human beings are capable of processing language accurately and efficiently without too much effort.

Up to this very moment, most computer systems that are capable of processing language use techniques that are unreliable, slow and non-trainable. In this thesis we present a number of biologically-inspired models that process language in a faster and more reliable manner. Moreover, most of the information these models will use, can be derived automatically. In other word, the models can learn. Although all models were experimental, they showed considerable relevance for applications in natural language processing and information retrieval.

In addition, I hope to take away some of the magic that surrounds the field of neural networks, and make a clear statement that these elegant and simple computing devices are nothing more than well understandable classification and categorization machines.

Here, it is argued that successful natural language processing systems are those systems that are based on the retrieval of analyzed language samples. These large collections of language (the *corpus*) need to contain well chosen, structured examples that are embedded in their natural context. This in contrast to systems that inference over large sets of rules over and over again, even for the most simple cases.

As an example, there exist many real-time applications in which it is impossible to inference knowledge within the required time-limits. One must immediately provide a good solution (which should not be the best, but just good enough). In these cases, corpus based techniques, as described in this thesis, are currently the only availible methods to implement such a functionality. However, one should never forget to implement those features

in a corpus-based system that could well handle properties of natural language such as hierarchical structures, ambiguity and lots of noise. It is exactly this problem that I have tried to address in this work.

This research could not have been realized without the trust and confidence of all the customers of M.S.C. Beheer B.V., who allowed me to combine the benefits of one's own business with those of a free-lance student. It was this confidence that allowed me to visit foreign conferences and universities, buy relevant literature and purchase the necessary computer facilities for the research presented here.

Moreover, institutions and companies such as the A.A.A.I., I.J.C.N.N., I.N.N.S., I.E.E.E., the University of Rochester, Koninklijke Shell Petroleum Maatschappij N.V., the Department of Computational Linguistics and the Faculty of Arts at the University of Amsterdam donated contributions, which I gratefully acknowledge. In addition, I would like to thank my collaborators at M.S.C., the members of the DIPDOP group, and my students for their contributions to the work presented, in particular Siebe Bloembergen, Lennert Hoogvliet and Martijn Lodewijks who implemented parts of this work.

This thesis appears in this form due to the valuable suggestions of Karin Bjarnason and my committee consisting of Prof. Dr. Ir. R.J.H. Scha, Prof. Dr. R. Bartsch, Prof. Dr. Y. Kamp, Prof. Dr. Ir. A. Nijholt, Dr. W. Daelemans, and Dr. Ir. B. Kröse. Finally I am greatly indebted to Prof. Dr. Ir. R.J.H. Scha whose ideas and patience were a source of inspiration for me and whom I could always rely on over the last four years.

I thank my parents and all other family and friends that have taken part in the realization of this work. Last but not least I am more than grateful to Frédérique who gave me a lot of support while I carried out this research. Much of this work is as much hers as it is mine.

Introduction

1.0 Machine Intelligence

For over fifty years the two main directions in machine intelligence (MI), neural networks (NN) and artificial intelligence (AI), have been studied by various persons with many different backgrounds. NN and AI seemed to conflict with many of the traditional sciences as well as with each other. The lack of a long research history and well defined foundations has always been an obstacle for the general acceptance of machine intelligence by other fields.

At the same time, traditional schools of science such as mathematics and physics developed their own tradition of new or "intelligent" algorithms. Progress made in the field of statistical reestimation techniques such as the Hidden Markov Models (HMM) started a new phase in speech recognition. Another application of the progress of mathematics can be found in the application of the Kalman filter in the interpretation of sonar and radar signals. Much more examples of such "intelligent" algorithms can be found in the statistical classification en filtering techniques of the study of pattern recognition (PR).

Here, the field of neural networks is studied with that of pattern recognition in mind. Although only global qualitative comparisons are made, the importance of the relation between them is not to be underestimated. In addition it is argued that neural networks do indeed add something to the fields of MI and PR, instead of competing or conflicting with them.

2.0 Natural Language Processing

The study of natural language processing (NLP) exists even longer than that of MI. Already in the beginning of this century people tried to analyse human language with machines. However, serious efforts had to wait until the development of the digital computer in the 1940s, and even then, the possibilities were limited. For over 40 years, symbolic AI has been the most important approach in the study of NLP. That this has not always been the case, may be concluded from the early work on NLP by Harris. As a matter of fact, Chomsky's *Syntactic Structures* was an attack on the lack of structural properties in the mathematical methods used in those days. But, as the latter's work remained the standard in NLP, the former has been forgotten completely until recently. As the scientific community in NLP devoted all its attention to the symbolic AI-like theories, the only useful practical implementation of NLP systems were those that were based on statistics rather than on linguistics. As a result, more and more scientists are redirecting their attention towards the statistical techniques available in NLP. The field of connectionist NLP can be considered as a special case of these mathematical methods in NLP.

More than one reason can be given to explain this turn in approach. On the one hand, many problems in NLP have never been addressed properly by symbolic AI. Some examples are robust behavior in noisy environments, disambiguation driven by different kinds of knowledge, commensense generalizations, and learning (or training) abilities. On the other hand, mathematical methods have become much stronger and more sensitive to specific properties of language such as hierarchical structures.

Last but not least, the relatively high degree of success of mathematical techniques in commercial NLP systems might have set the trend towards the implementation of simple, but straightforward algorithms.

In this study, the implementation of hierarchical structures and semantical features in mathematical objects such as vectors and matrices is given much attention. These vectors can then be used in models such as neural networks, but also in sequential statistical procedures implementing similar characteristics.

3.0 Information Retrieval

The study of information retrieval (IR) was traditionally related to libraries on the one hand and military applications on the other. However, as PC's grew more popular, most common users loose track of the data they produced over the last couple of years. This, together with the introduction of various "small platform" computer programs made the field of IR relevant to ordinary users.

However, most of these systems still use techniques that have been developed over thirty years ago and that implement nothing more than a global surface analysis of the textual (layout) properties. No deep structure whatsoever, is incorporated in the decision whether or not to retrieve a text.

There is one large dilemma in IR research. On the one hand, the data collections are so incredibly large, that any method other than a global surface analysis would fail. On the other hand, such a global analysis could never implement a contextually sensitive method to restrict the number of possible candidates returned by the retrieval system. As a result, all methods that use some linguistic knowledge exist only in laboratories and not in the real world. Conversely, all methods that are used in the real world are based on technological achievements from twenty to thirty years ago.

Therefore, the field of information retrieval would be greatly indebted to a method that could incorporate more context without slowing down. As computers are only capable of processing numbers within reasonable time limits, such a method should be based on vectors of numbers rather than on symbol manipulations. This is exactly where the challenge is: on the one hand keep up the speed, and on the other hand incorporate more context. If possible, the data representation of the contextual information must not be restricted to a single type of media. It should be possible to incorporate symbolic language as well as sound, pictures and video concurrently in the retrieval phase, although one does not know exactly how yet...

Here, the emphasis is more on real-time filtering of large amounts of dynamic data than on document retrieval from large (static) data bases. By incorporating more contextual information, it should be possible to implement a model that can process large amounts of unstructured text without providing the end-user with an overkill of information.

4.0 The Combination

As this study is a very multi-disciplinary one, the risk exists that it remains restricted to a surface discussion of many different problems without analyzing one in depth. To avoid this, some central themes, applications and tools are chosen. The themes in this work are self-organization, distributed data representations and context. The applications are NLP and IR, the tools are (variants of) Kohonen feature maps, a well known model from neural network research.

Self-organization and context are more related to each other than one may suspect. First, without the proper natural context, self-organization shall not be possible. Next, self-organization enables one to discover contextual relations that were not known before.

Distributed data representation may solve many of the unsolved problems in NLP and IR by introducing a powerful and efficient knowledge integration and generalization tool. However, distributed data representation and self-organization trigger new problems that should be solved in an elegant manner.

Both NLP and IR work on symbolic language. Both have properties in common but both focus on different features of language. In NLP hierarchical structures and semantical features are important. In IR the amount of data sets the limitations of the methods used. However, as computers grow more powerful and the data sets get larger and larger, both approaches get more and more common ground. By using the same models on both applications, a better understanding of both may be obtained.

Both neural networks and statistics would be able to implement self-organization, distributed data and context in the same manner. In this thesis, the emphasis is on Kohonen feature maps rather than on statistics. However, it may be possible to implement many of the techniques used with regular sequential mathematical algorithms.

So, the true aim of this work can be formulated as the understanding of self-organization, distributed data representation, and context in NLP and IR, by in depth analysis of Kohonen feature maps.

Chapter 1

Neural Computation

"When two elementary brain-processes have been active together or in immediate succession, one of them, on re-occurring, tends to propagate its excitement into the other."

"The amount of activity at any given point in the brain cortex is the sum of the tendencies of all other points to discharge into it, such tendencies being proportionate (1) to the number of times the excitement of each other point may have accompanied that of the point in question; (2) to the intensities of such excitements; and (3) to the absence of any rival point functionally disconnected with the first point, into which the discharges might be diverted."

-- William James, 1890

Abstract

In this chapter, neural networks and neural processes are introduced. Besides the artificial models such as back-propagation and Kohonen feature maps, general characteristics of biological neural networks are discussed. Moreover, facts known from aphasia and neuro-linguistic research are used to advocate a parallel distributed knowledge representation in artificial intelligence and natural language processing in particular.

Throughout the chapter, many historical anecdotes of the already four decades old neural research direction can be found, so the reader can understand the sensation surrounding the neural research in the proper context.

1.0 Introduction

In recent years, the human brain and neural networks received a tremendous amount of research attention. This chapter focuses on the details of this hype and on general aspects of neural computation. It is organized as follows:

First, the problems in symbolic Artificial Intelligence and the appeal of neural networks shall be discussed, explaining *why* people made the effort to investigate neural networks in the first place.

Computing neural networks are, in many aspects, related to their biological counterparts. In order to understand this relation properly and to judge artificial neural networks in their correct context, biological nervous systems shall be discussed next.

In fact, the general term "neural networks" cover a large collection of different computing devices. By using a classification scheme that is based on the dimensions: connection, neuron and learning-rule, the reader is provided with an impression of the different types of artificial neural networks.

From the early 40s up to the late 60s a large group of researchers worked with neural networks. As this work is important to understand the value of the current research, the next sub-sections focus on the fundamentals of neurocomputing and on models with imaginative names like Mark I Perceptron and ADALINE.

Best known and largely responsible for the hype surrounding neural networks are two major streams within the neural network research. First, there is the back-propagation algorithm, as proposed by David Rumelhart and James McClelland[1]. Less known, but just as important, is the work done by Teuvo Kohonen on self-organizing feature maps at the University of Helsinki. These two algorithms are discussed in the subsequent sub-sections.

Never in the history of computer science has a research topic been surrounded by so many emotionally loaded discussions and debates as the neural network paradigm. These 'new' processing elements seemed to conflict with the statistical pattern recognition tradition as well as with the artificial intelligence establishment. Terms like "paradigm shift" and "new definitions of rationality" were used throughout the discussions. Maybe one can understand this sensation better if one has a detailed insight in the rich history of the neural paradigm. Therefore, historical facts about neural network research are given throughout this chapter.

1. The back-propagation algorithm was invented simultaneously by different people [Le Cun, 1986], [Parker, 1985] [Werbos, 1974]. However, it were Rumelhart and McClelland who named this algorithm "back-propagation" and who related it to the problem of multi-layer perceptrons.

2.0 Problems with Symbolic Artificial Intelligence and NLP

Current symbolic Artificial Intelligence (AI) models suffer from many unresolved problems due to the limitations of the architecture and due to the knowledge representation schemes used[1]. The most important of those problems are:

- To resolve ambiguities, different knowledge sources must be consulted at the same time. In symbolic AI, these sources are normally separated in different system modules. Therefore, complex control mechanisms are needed to integrate them properly. Moreover, every system designed to work with separated modules contains communication channels between them. Explicitly defined communication channels between modules, show up as bottle-necks in the resulting system.
- In addition, as soon as some input value is slightly corrupted or disturbed by environmental noise, the entire system collapses. This inability to reason with incomplete or imprecise information is caused by the local data representation schemes that can be found in symbolic AI. These systems use memory models where *one* concept is stored in *one* particular slot. As soon as the slot is damaged or the input element changes a little, it can no longer be found in the memory bank, resulting in a system failure. Due to this strictly local data representation, generalization and error-correction are impossible[2].
- Most important and so far not explained by symbolic AI is the fact that humans have an adaptive way of learning. In symbolic AI, one must pre-code and pre-categorize real-world data into groups of symbols, before one actually starts to define relations between them. This results in problems in the long run. A system which is capable of discovering categories by itself should be preferred.
- Symbolic AI systems tend to be slow. On the one hand this is caused by the inability of current computer systems to process symbolic information. Vector representations can be processed much better. On the other hand, it is virtually impossible to parallelise symbolic AI techniques due to the strongly sequential nature of these techniques.

The next sections will discuss these problems in more detail as they occur in symbolic computational linguistics. At the end, a motivation for the use of connectionist methods shall be given.

1. There is a rich tradition in symbolic Machine Learning. In addition, there is much interest towards the field of symbolic Machine Learning [Daelemans et al., 1992]. However, the results achieved have mainly been of theoretical importance. There are no hard, practical applicable results yet.

2. The difference between local and distributed data representations shall be discussed more thoroughly further on in this chapter.

2.1 Ambiguity

In spite of the fact that ambiguity has always been one of the most important problems in natural-language processing and natural-language understanding, it was rarely solved by symbolic NLP. One can understand that determining the correct meaning of a word is one of the basic functions of a NLP system. In a way, the philosophy behind sequential-symbolic NLP-system architectures caused the lack of an efficient disambiguation model.

Different types of lexical ambiguity can be distinguished. Syntactic-lexical ambiguity: correct categorization of a word, e.g., noun verses verb. There are two types of semantic ambiguity. The first one is called polysemy: words with several meanings that are related ("The government fell" versus "John fell and hurt himself"). The second, homonymy, pertains to unrelated words with the same form ("foot-ball" versus "dance-ball"). To resolve these types of ambiguity, NLP systems must take various sources of knowledge in account.

Because most NLP systems work sequentially (first syntactical then semantical analysis), it is very difficult to solve the ambiguity problem. One cannot resolve the complete ambiguity in the syntactical part without taking into consideration the semantical disambiguation.

The same argument holds for the semantical phase. If there is no ambiguity resolved in the syntactical phase (i.e. all the various meanings are passed to the semantical phase), the semantical disambiguation definitely becomes too complex. So, there is a need for a method which solves the ambiguity problem by taking to account all the available knowledge at once.

Blackboards [Hayes-Roth, 1985], backtracking [Woods, 1970], delay [Marcus, 1980] and marker-passing [Charniak, 1983] tried to solve the problem by integrating different knowledge sources. However, the bandwidth between the different sources seemed to be broader than expected. Combining syntactical and semantical knowledge in one distributed knowledge base might do better.

2.2 Robustness

Humans often use language incorrectly; still one understands what someone else means if one communicates with him or her. Errors are made at different levels such as spelling, grammar and meaning. Because current NLP systems do not work with content-addressable memory and association (at all the levels) it is very difficult for symbolic methods to recognize and correct an incorrectly used word. The correction of a misspelled word in a non-associative memory system is an $O(n^3)$ problem, with n the number of possible words [Wagner et al., 1974]. This complexity can be brought back to $O(n^2)$ by using extra memory, but it remains complex. Solving a syntactical or semantical error is even worse.

Because of their architectural shortcomings, sequential NLP systems with classical memory utilization schemes are unable to solve these problems in an elegant way. Whenever a NLP system fails to work because one word has been entered wrongly into the system, this might be called a basic shortcoming.

2.3 Learning and Generalization

One of the disadvantages of symbolic NLP systems is their inability to learn automatically. Everything has to be encoded by hand: lexicon, syntax rules and semantical issues. Once the system encounters an unknown word combination, grammatical use or meaning, it does not know what is meant by it. Several researchers tried to solve this problem. However, symbolic methods seemed to be too complex for *efficient* learning algorithms [Kodratoff et al., 1990], [Michalski et al., 1986a, 1986b] [Winston, 1975].

Related to learning is the aspect of generalization. Once a system has certain information, it might be able to derive the meaning of unknown language use by generalizing. Therefore, high level integration of all the system's knowledge is needed, the architectural limitations of symbolic-sequential NLP systems do not meet this demand. This might be the reason why these systems still lack the ability to generalize. The term learning can also be positioned in a more general cognitive context: the problem of language acquisition.

The symbolic AI community never quite tackled the problem of language acquisition, probably the most interdisciplinary study in cognitive science.

2.4 Complexity

The complexity of natural language has forced researchers to split the problem into subproblems. The nature of sequential computers made them decide to use sequences of modules which interact at different levels. However, the interaction between the different partial solutions was too intense, so the systems were suffering from communication bottlenecks (physically as well as conceptually).

The hope that future computers would be more powerful, gave developers of the computationally complex solutions an excuse and hope to proceed. Although computers became more powerful than anybody expected, the complexity problem is still not solved. Therefore a theory of language processing is needed that posits, instead of simple passing of semi-complete results between processing parts, strong (parallel) interaction between those components.

Much used in symbolic NLP systems are rule-based knowledge representation methods. In the eighties these methods were mainly used because of their flexibility and expressive power. It turned out that the flexibility led to rule bases which are difficult to maintain, in particular when they are big. Moreover, it was difficult to split certain problems into separate sub-problems (needed for better maintenance, reduction of complexity and easier parallelization). Therefore, many times, rule based systems resulted in spaghetti-bases.

2.5 The Motivation for Neural Networks in NLP

So, if symbolic AI and NLP systems still suffer from such fundamental problems after so many years of research, one may question their abilities to model problems involving these forms of intelligent behavior. Therefore, it may be useful to study other computing models such as artificial neural networks because:

- A distributed representation might increase the system's ability to integrate different sources of knowledge in an elegant and effective way. By seriously integrating different kinds of knowledge, perception and common sense have a better chance of being achieved.
- Neural networks can correct corrupted real-world input. An even more interesting aspect of neural networks is the ability to generalize over known knowledge. Hereby, the system can react properly to unknown input.
- Directly following from the architecture is the computational power of a neurally inspired system. Natural language understanding has been shown to be an NP-complete problem. Connectionist models provide a solution to the computational complexity by enabling highly parallel computations.

Presumably most important is the argument that people do not have a store of knowledge (i.e. tables, addresses and pointers), people *are knowledge*. Neural networks provide a framework to implement such a memory-based model.

3.0 Biological Neural Networks

3.1 Background

Many reasons can be given why one would be attracted to biological neural networks as an inspiration source for building computing devices. For some the appeal of these elements is caused by their simple and elegant functioning. For others this appeal might be caused by the implicit massive parallelism or the built-in generalization capabilities. Moreover, neural networks show a robust and adaptive behavior that makes them attractive in noisy and changing environments. Whatever is the real reason, these simple computing devices already caused a research hype twice. Therefore it is interesting to analyse the working of their biological counterparts.

Biological neural networks consist of many simple computing elements. A neuron implements a non-linear threshold function. If the total of all input signals exceeds a certain threshold, the neuron fires (generates a voltage or potential *spike*). This output signal then serves as input for another part of the nervous system (which can be a muscle as well as another neuron). By combining billions of such neurons in a neural network, complex calculations and pattern comparisons can be carried out. The fact that each neuron fires with a maximum frequency of one thousand spikes per second, and that human beings can perform complex pattern recognition tasks within a tenth of a second, yields the conclusion that even complex calculations, like observing a ball in the air and reaching out to catch it, must be performed within 100 sequential steps. So, the network must use massively parallel computations in order to process all the information within this limited time. This so-called *100 step* rule was suggested by [Feldman et al., 1982]. This insight stimulated many psychologists to use neural networks in their modelling.

However, the artificial neural networks that are currently in use are often *very* simple abstractions of biological neural networks. Therefore, most of the artificial neural networks differ so much from biological neural networks that properties such as adaptive and robust behavior, implicit generalization and built-in parallelism, can no longer be expected to be necessarily present. To understand the limitations of such abstract models better, this section discusses the properties of biological neural nets.

In the section "Radical Connectionism and the Psychological Plausibility of Artificial Neural Networks" on page 60, four essential properties of artificial neural networks shall be mentioned. It is claimed, that if a model does not have these four properties, one cannot apply the properties of biological neural networks to it.

On the other hand, the speed, stability and accuracy of artificial (neural) components reaches far beyond that of biological neurons, indicating interesting new possibilities for these computing devices.

3.2 Neural Building Blocks

3.2.1 Nerve and Glial Cells

The three basic elements of a biological neural network are neurons, binding cells and connections. Neurons transfer information by exchanging electrical signals. These electrical signals are caused by potential differences between chemical elements such as Calcium, Kalium, Natrium and Chloride (Ca^{2+}, K^{+}, N^{+}, Cl^{-}). Human beings have about 10^{12} neurons and 10^{16} interconnections. These nerve cells can be classified into 1,000 to 10,000 different groups, depending on the features used to classify them. The binding cells are called *Glial* cells. There are about 10 to 50 times more glial cells than neurons. These cells do not generate electrical activity and their main function is to keep the neurons in place and to protect them from damage by external factors.

Each neuron is constructed of four basic elements: the neural *soma*, some input *dendrites*, input *synapses* and an output *axon*. The soma is a cell that implements a threshold function and generates electrical signals. On the one hand, the soma receives inputs from dendrites which are connected to the soma through synapses. On the other hand, the soma generates electrical output transmitted through the axon: a tube that can reach lengths up to one meter in human beings. The axon has several interruptions, called the *Nodes of Ranvier*, in honor of Louis Antoine Ranvier, who discovered them at the end of the 19th century. At the end, the axon splits into *terminal fibres*, that hook up to other parts of the nervous system (see figure 1).

What is currently known from biological neural networks is largely due to Camillo Golgi. In 1873, Golgi invented his *Silver Impregnation* method that enabled him to analyse the structure of an individual neuron by using a microscope. This method has two important properties. First, it stains only one to ten percent of all the cells in a particular brain region, making it possible to study a neuron in relative isolation. Second, neurons that take up the stain, are complete neurons, including axon and dendrites [Golgi, 1906].

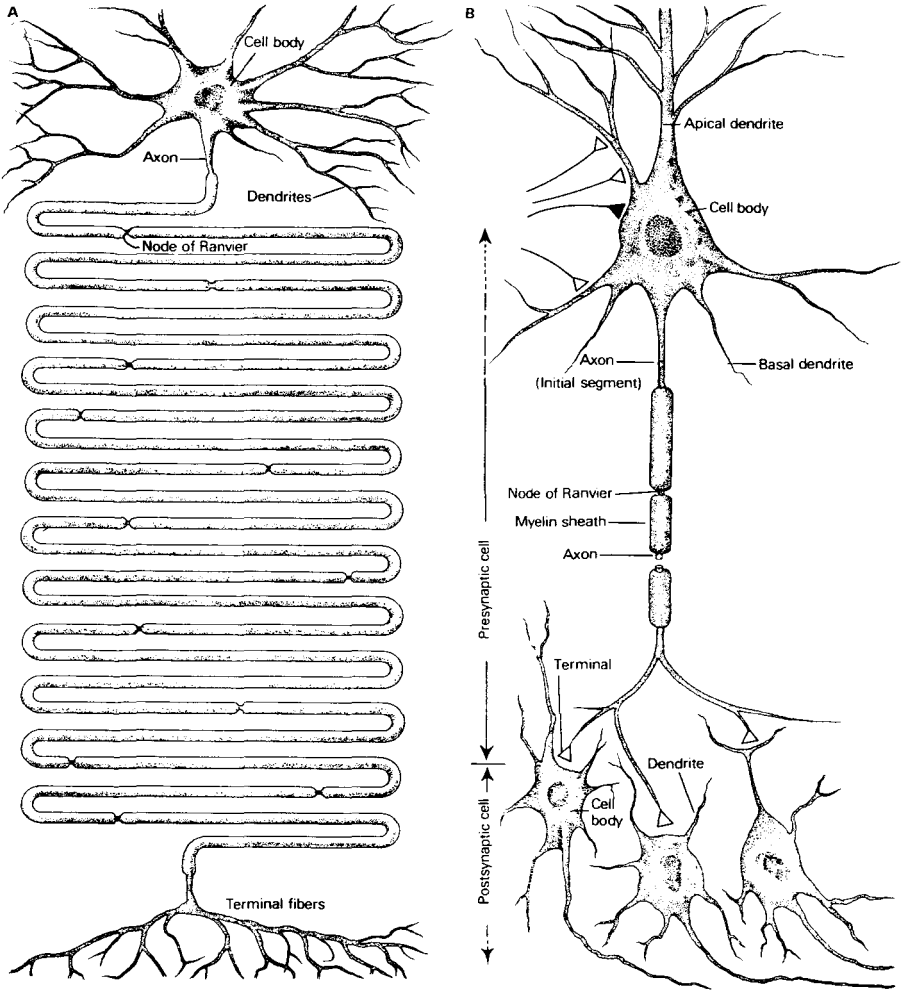


FIGURE 1. A typical neuron. On the left side of the figure one neuron with its cell body, dendrites, and axon is presented. On the right hand side of the picture, a neuron and the connections with some other neurons is shown (reprinted from [Kandel et al., 1985]).

Using Golgi's method, Santiago Ramon y Cajal (who shared the sixth Nobel prize for Medicine with Golgi in 1906) showed that the human nervous system is not just a bunch of randomly connected individual neurons, but a highly structured network. Moreover, Cajal derived two important insights that still form the basis for today's cellular connectionist theory. First, Cajal defined the *principle of dynamic polarization*, stating that the information flow in neurons is one-directional. Next, he defined the *principle of connectional specificity* stating that neurons only exchange information through axons and den-

drites, that nerve cells do not form random networks, and that each cell has specific connections that link it with some cells and not with others. Cajal analyzed almost every part of the human cortex with Golgi's method, discovering thousands of different types of neurons [Cajal, 1906].

Compared to the silver impregnation method, earlier researchers were blind. However, nowadays, advanced scanning methods enable us to analyse global and regional brain activities, something Cajal was not capable of. By studying these patterns carefully, interesting new properties of the brain are being discovered.

Nerve cells can be categorized in three major groups (depending primarily on their shape): unipolar, bipolar and multipolar nerve cells.

- In unipolar nerve cells, one primary process gives rise to many branches. There are axon branches and dendrite branches. These cells are mainly found in invertebrates. However, a variant can also be found in the dorsal roots of vertebrates.
- Bipolar cells have one dendrite and one axon. The dendrite may be any collection of input from different cells; there is only one synaptic connection. The dendrite mostly picks up information from some sensor. The axon then transports it to the nervous system. Bipolar cells can be found in the retina.
- Multipolar cells are found mostly in vertebrate nervous systems. These cells are characterized by multiple dendrites and one axon that is connected to many other nerve cells. The *spinal cord* cell, the *pyramidical* cell and the *Purkinje* cell are some multipolar nerve cells. The spinal cord cell has about 10,000 dendrite inputs. However, the Purkinje cell of the cerebellum can contain up to 150,000 contacts. The pyramidical cell is a cell typically found in the human cortex (see figure 2).

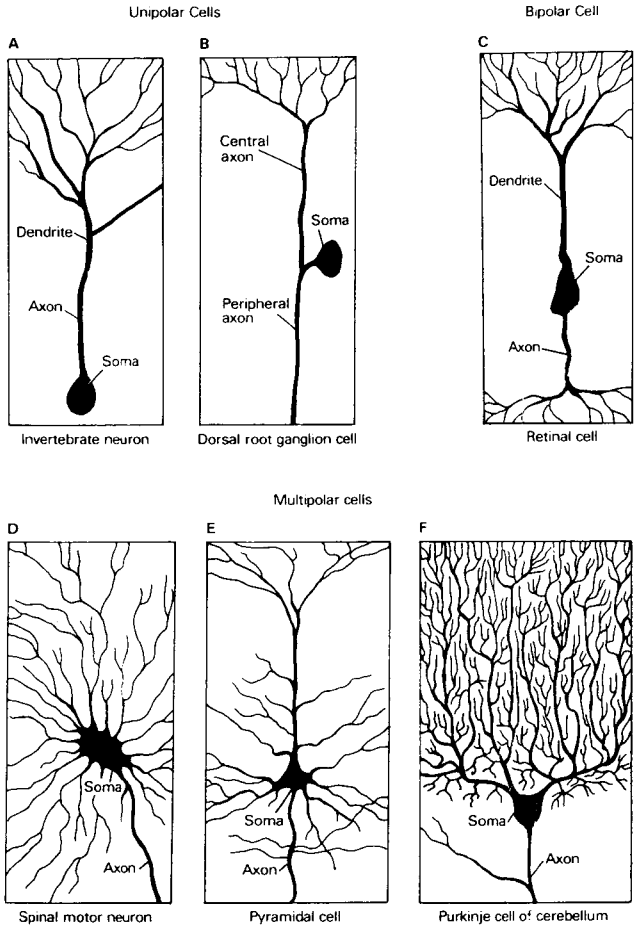


FIGURE 2. Different neuron groups. Unipolar, bipolar and multipolar cells from different parts of the human nervous system (reprinted from [Kandel et al., 1985]).

3.3 Biological Activation Functions

Whenever a neuron is triggered by its input activations, it might fire (or get activated) by sending an activation spike over its axon. These spikes differ much for different types of neurons, however a typical behavior can be extracted (see figure 3, "Typical potential spike of a biological neuron in time.").

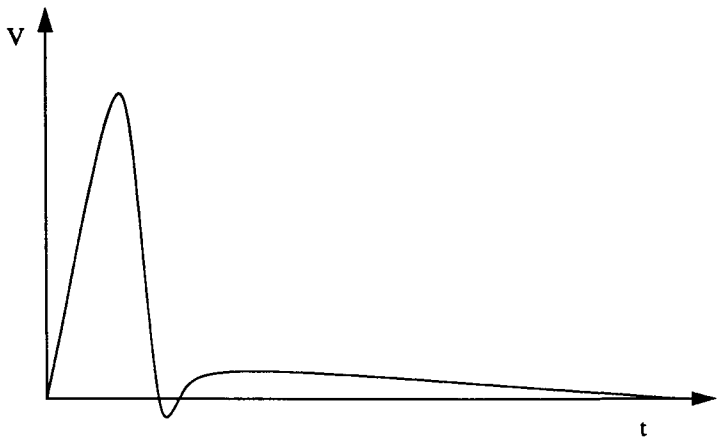


FIGURE 3. Typical potential spike of a biological neuron in time.

During the first few milliseconds, the potential increases rapidly. Next, after the potential has reached a peak, it falls back towards a negative value and oscillates a couple of times, after which it slowly reaches a zero value. This function is non-linear and behaves rather differently than the functions implemented by the neurons of current artificial neural networks.

3.4 The Human Nervous System

Before the human central nervous system can be described in more detail, some neuroanatomical terms have to be understood. In human anatomy the words *rostal, caudal, dorsal,* and *ventral* are used in order to express the directions in which neuroanatomical elements are ordered in relation to each other. The brain and the spinal cord are organized along two axes. One is the rostral-caudal axis from nose (Latin, rostrum) to tail (Latin, caudal). The other is the dorsal-ventral axis that runs from back (Latin, dorsum) to the abdomen (Latin, venter). Originally these axes where derived with animals in mind. However, during human evolution, the nervous system bended above the brain stem. So, the actual meaning of the axes depends on the part of the nervous system one discusses (see figure 4).

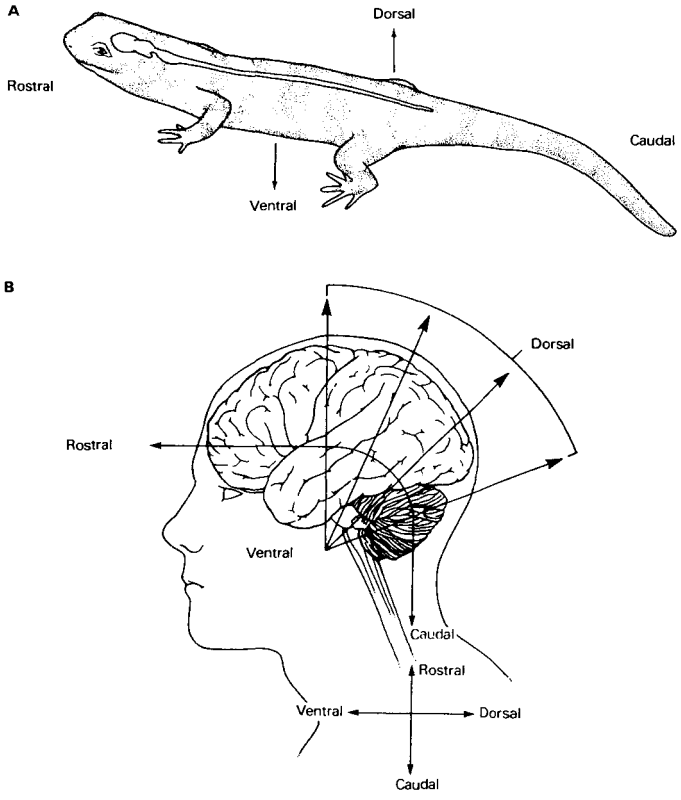


FIGURE 4. The axes of the central nervous system. A. With the lower vertebrates, the nervous system follows a straight line. B. In the human nervous system a bending occurs above the brain stem, resulting in a redefinition of directions (reprinted from [Kandel et al., 1985]).

The human nervous system shows an extremely symmetrical structure. Overall, it consists of six main parts (see figure 5).

- the *spinal cord* transports information from all over the body to the brain. This part is the most caudal one of the nervous system.

- the *medulla oblongata* is rostral with respect to the spinal cord.
- the *pons* and the *cerebellum* are rostral with respect to the medulla. All movements and balance issues are taken care of by the cerebellum.
- the *midbrain*, also referred to as the *brain stem*, lies between the *hindbrain* (medulla, pons and cerebellum) and the *forebrain* (diencephalon and cerebral cortex).
- the *diencephalon* is the information preprocessor of the cerebral cortex. It consists of two main parts. The thalamus and the hypothalamus. The first actually processes the information for the cerebral cortex. The second is involved in information integration.

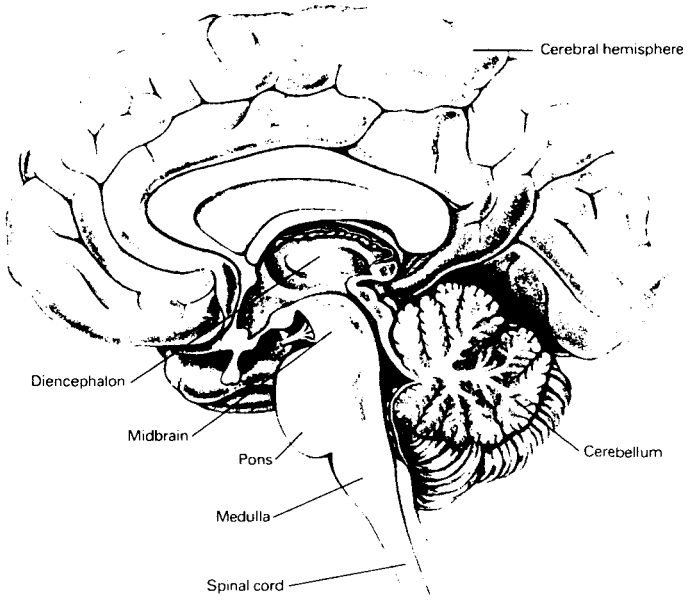


FIGURE 5. The central nervous system (reprinted from [Kandel et al., 1985]).

- the *cerebral hemispheres* contain the *basal ganglia* and the *cerebral cortex*. The higher level motor, cognitive, and perception functions are concentrated in this part of the central nervous system.

For the rest of this chapter emphasis will be on the cerebral cortex, because this part of the nervous system is mainly involved in cognitive functions such as language.

The cerebral cortex is a large collection of nerve cells (or neurons) and glial cells. The cerebral cortex is folded in the human skull (folding is an old trick of nature to increase surface area). Unfolded, the cortex is the size of a napkin and is some millimeters thick. The cortex consists of six layers. These layers are organized in columns. In each layer, specific types of neurons exist. Information processing takes place from the lower levels up to the higher levels. The lower layers are involved in local sensory information pre-processing, where the higher layers are specialized in association tasks and long distance inter-neural communications (see figure 6). There are indications that these cortical columns are involved in serial processing.

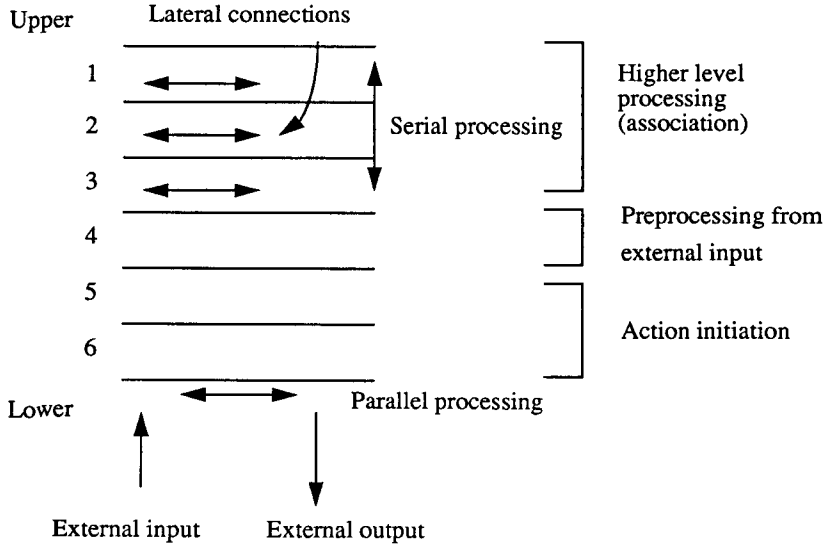


FIGURE 6. Cortical layers and cortical columns.

The cortex is organized in two *hemispheres*. Each of these two hemispheres can be divided into four lobes: the *frontal*, *parietal*, *occipital*, and *temporal*.

In general, the brain uses a locally distributed data representation. This means that specific regions are responsible for specific actions, but that within such a region, a distributed data representation is used. If one studies the four lobes of the cerebral cortex, the frontal lobe is mainly involved in planning and movement. The parietal lobe takes care of somatic sensations. Vision is arranged by the occipital lobe, and finally the temporal lobe is responsible for audition, learning, memory and emotion. Each lobe has a primary sensor part and an higher order part. Moreover, the areas in between these four distinct areas are called *association* areas. These areas are mainly responsible for the integration of information from different lobes i.e. the visual and the motor area (see figure 7).

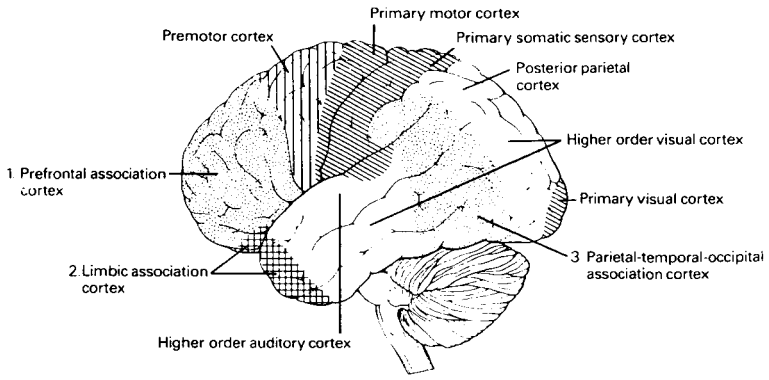


FIGURE 7. The medial and lateral surface of the human brain (reprinted from [Kandel et al., 1985]).

In studying the role of the two brain hemispheres, it is apparent that each one is responsible for the sensory and motor functions of the opposite side of the body. In addition, the two hemisperes are not completely symmetrical and they do have different functions. Much of what is known from the localization of specific functions in the brain is derived from the study of *aphasia*, in other words, the study of language disorders. Because of the

importance of these results for the study of neural networks and natural language processing, the next section is devoted to this subject.

3.5 Aphasia

In 1861, the French neurologist Pierre Paul Broca described a case involving one of his patients who could not speak grammatically. The patient could however sing a melody or speak isolated words, but combining the words into a grammatical sentence was beyond his capabilities. After the man died, Broca discovered a lesion (brain damage that may have been caused in many different ways) in the posterior portion of the frontal lobe (now called Broca's area). In the following years, Broca researched more of such cases. All of them showed lesions in the same brain region. As a result, Broca discovered that speech originated in the left hemisphere [1] [Broca, 1865].

In 1876 the German researcher Carl Wernicke discovered another form of aphasia caused by brain lesion. Broca's patients could understand but not speak. Wernicke's patients on the other hand could speak but not understand. This typical behavior was caused by lesions at the intersection of the posterior part of the temporal lobe and the parietal and occipital lobes [Wernicke, 1908].

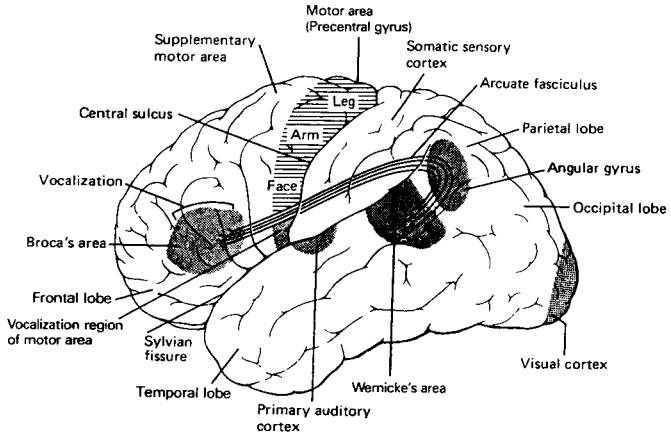


FIGURE 8. The primary language areas of the brain (reprinted from [Kandel et al., 1985]).

By combining the work of Broca and that of other researchers such as Gustav Theodor Fritsch and Eduard Hitzig, Wernicke formulated a theory in which he stated that basic

1. Or, as he stated it: "Nous parlons avec l'hemisphere gauche!".

mental functions could be localized in specific brain regions and the more higher level (complex) cognitive functions arise from interactions among those elementary perception and motor areas. Moreover, he stated that information processing in the brain is performed in parallel as well as serially. So, already at the end of the nineteenth century, Carl Wernicke was the first to use the notion of *parallel distributed processing*.

According to Wernicke, auditory and visual sensory information merge in the Wernicke region, so they can be recognized as written or spoken language. If there is a lesion in this area, language comprehension is not possible. In Broca's area, language takes on its spoken or written form. If there is a lesion in this part of the brain, one can no longer articulate language (see figure 8).

Wernicke used this model to predict another form of aphasia that was discovered later: *conduction aphasia*. If there were a lesion between Broca's and Wernicke's area, the patient would be able to understand language and speak fluently, however, he could not repeat simple sentences correctly. The patient would omit and substitute words. Moreover, the patient would be completely aware of the errors, but would be unable to correct them.

Although some different theories came up during the beginning of this century, Wilder Penfield's surgery on epilepsy patients confirmed Wernicke's theories on locally distributed data representation [Penfield et al., 1950].

4.0 Neural Network Models

4.1 Background

In general, the last fifty years of machine intelligence research can be characterized by the following global periods. In the 1940s people started to explore simple neural networks. These simple networks were all constructed by hand. There was no notion of learning whatsoever. In the 1950s, the focus was on learning. By automatically adapting the weights between the neurons, long term memory could be implemented in the machines. The beginning of the 1960s was dominated by the connectionist approach. Most important were mathematical and biological influences from cybernetics and neuropsychology. At the end of the 1960s the symbolic approach became increasingly popular. In the 1970s, focus was on the representation of knowledge; expert systems and knowledge based systems. The 1980s were under the spell of the revival of (neural) learning machines. As far as the 1990s have proceeded, a trend towards biological neural networks and statistical learning algorithms can be noticed.

The first people to implement a neural network were Marvin Minsky and Dean Edmonds in the summer of 1951. The memory of this machine (called the SNARK) was stored in the position of the control knobs. The machine was made of 40 such knobs. When the machine was learning, it changed the position of the knobs by using a B24 bomber gyropilot. Minsky was so amazed by the functioning of this machine, that it stimulated him to write his Ph.D. thesis on a problem related to machine learning. Moreover, this machine was so complex, that they could never debug it 100%. However, "its random design was almost sure to work no matter how you built it" [Bernstein, 1981].

4.2 The Basic Elements of an Artificial Neural Network

Four decades of research in neural networks provide us with more than fifty different neural network structures. Overall, it can be stated that a neural network is characterized by three main features: the distributed processing elements, the connections between them and the learning rule. Although all neural networks have this in common, many variations have been invented lately.

In [Lippmann, 1988] a good overview of the most popular algorithms and models of this confusing field is given in a clear and intelligible way.

4.2.1 Data Representation

In localist connectionism, each concept or hypothesis about the environment is represented by one unit. Relations are represented by two types of links: inhibitory and excitory, both having the same functionality as synapses in biological neural networks. Most

localist networks show a hierarchical structure. The disadvantage of local representations is the fact that it is quite hard to implement connectionist properties such as robustness and adaptive behavior. Therefore powerful heuristics and techniques need to be developed. There is a parallel between this approach and symbolic artificial intelligence, with all the problems discussed earlier.

Distributed connectionism on the other hand, is based on representing concepts as patterns over large numbers of units. Each unit represents a microfeature of the pattern. Similar concepts share similar microfeatures and therefore have similar patterns. Distributed connectionism provides an efficient way to represent knowledge. Moreover, it provides us with a self-generalizing, associative representation. Mainly because of this last reason, distributed representations must be preferred above local ones [Feldman et al., 1982].

4.2.2 Activation Functions

Apart from the general structure of the neuron, an activation function must be defined. That is, the exact manner a neuron reacts to input activations. Biological activation functions have very non-linear characteristics. Artificial activation functions are very simple abstractions of such functions. The activation (firing rate) of a neuron is a function of the network activity. Each firing event is called a *spike*. In mathematical terms, an activation x_i of neuron i as a function of the activity of the network elements that are directly connected to neuron i: net_i, can be written as:

$$x_i = f(net_i) \qquad \text{(EQ 1)}$$

where w_{ij} is the weight of the connection between neuron i and j, x_j is the output activation of neuron j, and net_i is the activity received by neuron i from the entire network.

In general, three different types can be distinguished: the *linear*, the *threshold*, and the *sigmoid* activation functions. First the linear:

$$f(net_i) = \sum_j w_{ij} x_j \qquad \text{(EQ 2)}$$

The *linear* activation function is the simplest one. In this case:

$$x_i = f(net_i) = c \cdot net_i \qquad \text{(EQ 3)}$$

where c is a constant, also called the *gain* of the activation function (see figure 9, "Linear activation function"). c is the same for the entire network.

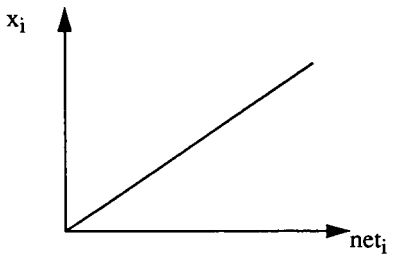


FIGURE 9. Linear activation function

A better approximation of a biological activation function is a *(relay) threshold* function. In this case, the neuron only fires at a certain input activation.

$$x_i = f(net) = \begin{cases} 0, (net_i < a) \\ 1, (net_i \geq a) \end{cases} \quad \text{(EQ 4)}$$

where a is the threshold value (see figure 10, "Threshold activation function"). This value is the same for the entire network.

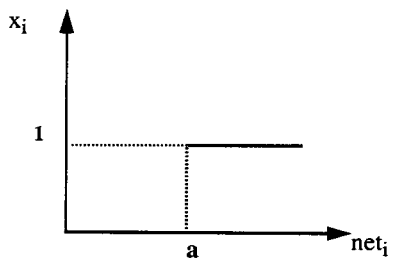


FIGURE 10. Threshold activation function

This function is non-linear and non-continuous, which causes great difficulties in mathematically analyzing the behavior of the model. Therefore, a variation can be found in the literature, the *semi-linear* threshold function. This function has a linear behavior between a minimal and a maximal value of the input activation. This function has the advantage that it approximates the biological activation functions well enough, while keeping a lin-

ear character. A disadvantage of the function is the fact that there are discontinuities in the first derivative.

$$x_i = f(net_i) = \begin{cases} 0, (net_i < a) \\ c \cdot net, (a \leq net_i \leq b) \\ 1, (net_i > b) \end{cases} \quad \text{(EQ 5)}$$

where a is a lower threshold, b an upper threshold and c a constant equivalent to the gain in the linear activation function. Sometimes, this function is referred to as a *two-level threshold* function (see figure 11, "Semi-linear threshold function").

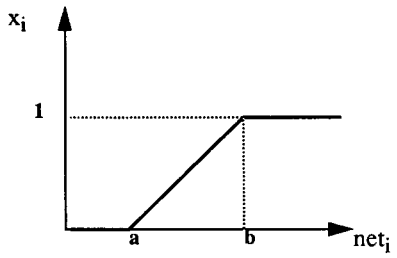


FIGURE 11. Semi-linear threshold function

The best non-linear function which is continuous in all derivatives, is the *sigmoid* function. This is a non-negative, everywhere monotonically increasing function that approaches zero and one respectively for $x \to -\infty$ and $x \to \infty$. The sigmoid function approximates the biological activation functions quite well.

$$x_i = f(net_i) = \frac{1}{1 + e^{-(\lambda \cdot net_i)}} \quad \text{(EQ 6)}$$

where λ is also called the *steepness* of the activation function. The problem with this function is the non-linear character. Therefore, it could not be applied until a training method was developed that could handle non-linear activation functions (see figure 12, "Sigmoid threshold function").

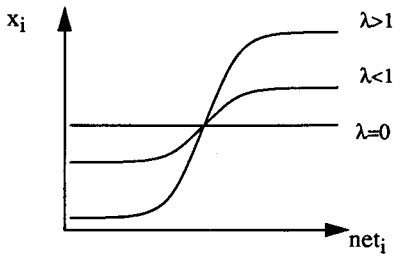


FIGURE 12. Sigmoid threshold function

The McCulloch & Pitts model uses a threshold activation function. The ADALINE and Perceptron model use linear activation functions. The non-linear sigmoid activation function is often applied in back-propagation neural network.[1]

4.2.3 Network Topology

Two main-stream network topologies can be distinguished: the *feed-forward* and the *recurrent* (or feed-back) neural networks.

- In feed-forward networks, the output activation is in one direction only: forward. The neural network has input activity on one side, then perhaps a number of in-between layers and then an output activity on the other side. There are no recurrent connections. Good examples of feed-forward networks are the Perceptron, ADALINE and the back-propagation network (also called multi-layer perceptron).
- Recurrent networks do have feed-back connections. This means that the output activation of the network at a certain time depends on the input, as well as on the output activation of the neural network. Therefore, recurrent neural networks have very complex dynamical properties. Some examples of recurrent neural networks are associative memories, the Hopfield model, Kohonen feature maps, ART (1 and 2) and the Simple Recurrent Network (SRN).

The McCulloch & Pitts model actually only defines a neuron. These neurons can be connected in any possible way. Therefore the McCulloch & Pitts model does not have a place in this classification.

1. More on the specific details of these models can be found in a later section of this chapter, after the basic principles such as activation rules, network topology, training rules and data representation are explained.

4.2.4 Learning Rules

Two different kinds of learning can be distinguished: supervised and unsupervised learning.

- In a supervised learning model, there is an explicit external teacher that provides the network with input-output pairs. The weights of the network are adapted by a function of the desired and the current output value for a specific input activation.
- Unsupervised models adapt the connection weights by taking into account the activations of the neighboring (clusters of) neurons. There is no external teacher and there is no predefined set of categories of which the input stimuli are part. The term *self-organization* is often used in relation with unsupervised models.

The supervised as well as the unsupervised learning rules are all derived from the famous *Hebb* learning rule, described by Donald Hebb in his 1949 work on synaptic learning: "The Organization of Behavior".

Hebb did not include any quantitative descriptions of synaptic adaptation, however, he was close enough when he stated:

> "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased".

The Hebb rule states that if two neurons i and j are activated simultaneously, the strength of the connection between them w_{ij} (weight) must be increased.

$$\Delta w_{ij} = \varepsilon \cdot x_i \cdot x_j \qquad \text{(EQ 7)}$$

where ε is the so-called learning rate and x_i and x_j are the activation values [Hebb, 1949].

Besides this very important rule, Hebb also proposed the idea of short term memory as a recurrent connection between (or within) neurons and the idea of distributed representations.

A real Hebbian way of learning can be found in [Malsberg, 1973] and [Linsker, 1988]. In these models, there are no restrictions to the interconnections of the model. All interconnections can be learned, the inter-layer as well as the one-layer interneuronal connections. Firing may be as non-linear as one wishes it to be, i.e. sigmoid, threshold or delay functions can be used. Although this model is biologically plausible and its results are promising, it requires a lot of computation and is therefore not popular at all within the already complex NLP applications.

The perceptron, ADALINE, and the back-propagation neural networks (supervised learning models), as well as associative memories, Kohonen feature maps, the Simple Recurrent Network (SRN), and ART (unsupervised learning models) use Hebbian learning rules.

4.3 The McCulloch & Pitts Neuron

Some say the neural network paradigm started with the famous McCulloch & Pitts paper in 1943. Others say the start goes back to the work of the psychologist William James in the 1890s. Whatever is right, neuron-like computer architectures have inspired researchers for many years. James' work suggested many of the neural network paradigms as they are known today (although he was not aware of the precise operation of these computation elements)[1]. In the late 30s and early 40s, other researchers like Norbert Wiener[2] and even John Von Neumann[3] (whose name is given to modern serial computer architectures with their so-called Von Neuman bottleneck) suggested brain-like research. However, the real breakthrough had to wait until the middle of the second world war.

In 1943, Warren McCulloch and Walter Pitts wrote "A Logical Calculus of the Ideas Immanent in Nervous Activity". In this paper they proved that the basic elements of the logical calculus: AND, OR and NOT could also be implemented in small, neural-like computing elements: the McCulloch & Pitts neuron. Therefore any finite logical expression could be realized by a McCulloch & Pitts network.

Each neuron has a fixed threshold. Depending on the input it receives through excitatory and inhibitory synapses, it fires a signal through the axon if the total input exceeds the threshold of the neuron. Although the networks had to be constructed by hand (learning was not possible), this work showed the possibilities of small computing neural networks in detail. Moreover, it had an immense influence on neuroscientists as well as on computer scientists [McCulloch et al., 1943].

A neuron was modeled as a *logical threshold unit.* A neuron has a number of binary input signals x, and one output signal y. Neurons can be linked together, so the output of one neuron is the input for the next. The output activation y is determined by a threshold function that fires if the addition of the input activities from neuron i, x_i, times the weights of the connection with neuron i, w_i, exceeds a certain threshold s.

1. James proposed a general elementary principle of association that is similar to Hebb's learning rule and he described the basic operations of neurons, calling them "points in the brain cortex" [James, 1890].

2. Norbert Wiener was the founder of the field of Cybernetics [Wiener, 1948/1961].

3. In 1945, Von Neuman wrote a draft research report in which he clearly states how a computer should store its information and its programs [Von Neuman, 1945/1982]. In the same report he discusses the work of McCulloch and Pitts in several places. The text shows that von Neumann was aware of the potential analogy between digital computers and the brain, even when he was designing serial computers.

$$y = \Theta\left(\sum_i w_i x_i - s\right) \quad \text{(EQ 8)}$$

where Θ is a threshold function of the form:

$$\Theta(x_i) = \begin{cases} 0, (x_i \leq 0) \\ 1, (x_i > 0) \end{cases} \quad \text{(EQ 9)}$$

It was shown by McCulloch and Pitts that any arbitrary logical function (such as AND, OR, NOT, XOR) could be constructed by a combination of such elements (see figure 13, "The McCulloch & Pitts logical threshold unit").

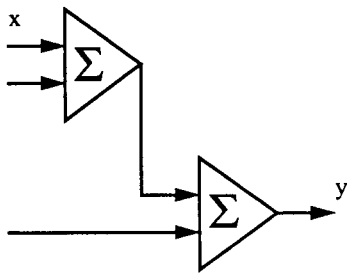


FIGURE 13. The McCulloch & Pitts logical threshold unit

Although McCulloch and Pitts started the neural networks research, they dealt with logical circuits rather than with neural networks in the current sense. Therefore they did not treat two important aspects:

- How one should train the weights. All weights in their models were hand-constructed.
- Biological neural networks show significant redundancy. This means that a number of neurons can be eliminated without influencing the functionality of the model too much. In the Logical Threshold Unit, one defect neuron directly influences the output activations. This problem is caused by the local data representation of the model.

4.4 The Perceptron

Frank Rosenblatt, a Bronx High School of Science classmate of Minsky, introduced a new phase in neural research with his 1958 paper on "The Perceptron: a Probabilistic Model

for Information Storage and Organization in the Brain". His invention (the "Mark I Perceptron") was important in more than on way. First, this was finally a computing device that *did* something. Originally, Rosenblatt was a psychologist, and so his perceptron did what a psychologist thought was important. Second, this was a machine that was capable of learning something, and that was what engineers wanted to put their hands on. Finally, although simple at first sight, perceptrons were mathematically extremely complex which made them interesting for mathematicians studying complex non-linear systems.

A perceptron is a supervised feed-forward network that uses binary (linear) neurons. Perceptrons performed remarkably well in pattern classification. In fact, perceptrons were taught to give transformations from an n-dimensional feature $\{0,1\}^n$ space to $\{0,1\}$ (see figure 14).

Rosenblatt used his perceptrons mainly to classify gray levels of image pixels into characters or shapes. This was primarily caused by the fact that he modelled the perceptrons according to biological nervous system structures he found in the retina [Rosenblatt, 1958].

Learning was implemented by a simple *reinforcement rule*. This rule could be self-organizing as well as supervised (or *forced adaptation* as Rosenblatt called it). The perceptron model consists of two layers: the *A* Units (on which the object is projected), and the *R* Units which classify the object. The *R* Units are randomly connected to the *A* Units. The weights between the *A* and the *R* Units can be changed. If an *R* Unit is activated at a certain moment in time and a connected *A* Unit is also activated, then the connection strength between them is increased.

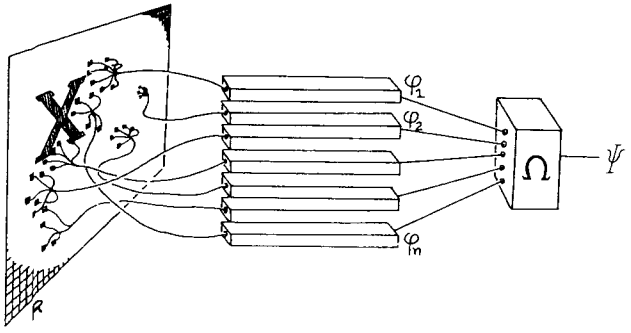


FIGURE 14. The Perceptron (reprinted from [Minsky et al., 1969/1988]).

Perceptrons use the following activation function:

$$y_j = \Theta\left(\sum_{i=1}^{n} w_{ij}x_i\right) \qquad \text{(EQ 10)}$$

where Θ is a so-called step function.

In 1962, Rosenblatt introduced his Perceptron convergence theory, in which he explains and proves the correctness of his perceptron learning rule [Rosenblatt, 1962]. The learning algorithm works as follows:

1. Provide all connections with random weights.
2. Select an input vector from the training set.
3. Feed-forward this vector. If the perceptron responds properly, do nothing. Otherwise, change the connections.
4. Continue with step 2 until the network classifies the training data correctly.

The most important result of this theory is the *Perceptron Convergence Theorem*, which states that:

> "If there exists a set of connection weights which is able to classify the input patterns in the corresponding classes, then the perceptron learning rule will converge to this set of weights in a finite number of steps, regardless of the initial set of weights."

The Mark I Perceptron was the first successful neurocomputer. It was trained to recognize characters from a 400 pixel image sensor. The Mark I Perceptron could store up to 512 weights. This computer appeared to be very successful in its task and the world expected (too) much from this machine. As Rosenblatt stated:

> "The question may well be raised at this point of where the perceptron's capabilities actually stop....the system described is sufficient for pattern recognition, associative learning, and such cognitive sets as are necessary for selective attention and selective recall. The system seems to be potentially capable of temporal pattern recognition....with proper reinforcement it will be capable of trial and error learning, and can learn to emit ordered sequences of responses."

Although another 30 years passed before some of these aspirations of perceptrons were fulfilled, they were not too ambitious.

To appreciate how enthusiastic Rosenblatt exactly was consider this phrase from [Rosenblatt, 1958]:

> "It seems clear that the class C perceptron introduces a new kind of information processing automaton: For the first time, we are having a machine which is capable of having original ideas. As an analogy of the biological brain, the perceptron, more precisely, the theory of statistical separability, seems to come closer to meeting the requirements of a functional explanation of the nervous system than any system previously proposed."

Less well known is the fact that Rosenblatt himself was also aware of the limitations of his computing devices; in particular in the case of relative judgements and symbolic behavior. In these cases, Rosenblatt described the behavior of his perceptrons as that of "brain damaged patients" [Anderson et al., 1988] [Hecht-Nielsen, 1990].

4.5 The ADALINE

The Perceptron Convergence Theory, as proposed by Rosenblatt, sometimes resulted in very long training times, and if finally a set of weights was found, it was not known whether it was the most optimal one.

In 1960, Bernard Widrow and Ted Hoff proposed a faster type of neuron learning algorithm. The computing device belonged to the family of perceptrons. It was called an "Adaptive Neuron" and was implemented by a Threshold Logic Unit with variable connection strength. Applied to the ADALINE, Widrow and Hoff were able to develop a supervised training algorithm for single layer neural networks: *the delta rule.*

Almost all of today's learning rules are derived from the delta rule. For instance, the back-propagation algorithm is a generalized case of the delta rule for multi-layer perceptrons with non-linear activation functions.

The ADALINE[1] was the first learning model for continuous signals. One of the main advantages of the ADALINE was that it could be implemented in hardware easily. The model uses linear activation functions:

$$y_j = \sum_{i=1}^{n} w_{ij} x_i + w_0 \qquad \text{(EQ 11)}$$

1. In its first days, this acronym stood for ADAptive LInear NEuron. Later it was changed to ADAptive LINear Element, as neural networks became less popular.

The problem with the Hebb rule is that the value of the weight keeps on increasing as the neurons i and j are activated. The delta rule suffers no longer from this problem. However, as a result, the delta rule is a supervised training rule, where the original Hebb rule is unsupervised. The delta rule follows the equation:

$$\Delta w_{ij} = \varepsilon (y_i^{(corr)} - x_i) x_j \qquad \text{(EQ 12)}$$

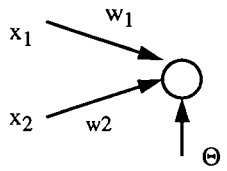
where ε is the learning rate, $y_i^{(corr)}$ is the desired activation for neuron i and x_i is the actual activation of neuron i.

Because the ADALINE could be easily generalized towards larger networks, Widrow and Hoff tried to develop a multi-layer supervised training algorithm for many years. Unfortunately, they did not succeed. The world had to wait for the back-propagation algorithm to implement such a functionality. Nowadays, everybody who owns a facsimile machine or a 2400 baud modem also owns an electronic implementation of a Widrow and Hoff "Adaptive Neuron" [Widrow et al., 1960].

4.6 The XOR Problem

By the end of the 1960s, the first golden period of neural network research ended with the book "Perceptrons, An Introduction to Computational Geometry" by Marvin Minsky and Seymour Papert. In this very important work they showed that the single layer perceptron could never learn the Exclusive-OR problem. In fact they proved that there is no (single layer perceptron) weight set for the XOR problem by three simple graphs (which were proceeded by an extensive geometric analysis of the behavior of perceptrons).

Assume there is a single layer perceptron with two inputs:



Then, the activity of the network can be calculated as:

$$y = w_1 x_1 + w_2 x_2 + w_0 \qquad \text{(EQ 13)}$$

For the AND, OR and XOR function, the input space can be represented geometrically in the following graphs. Open points indicate a zero as output of the perceptron, closed points indicate a one as desired output.

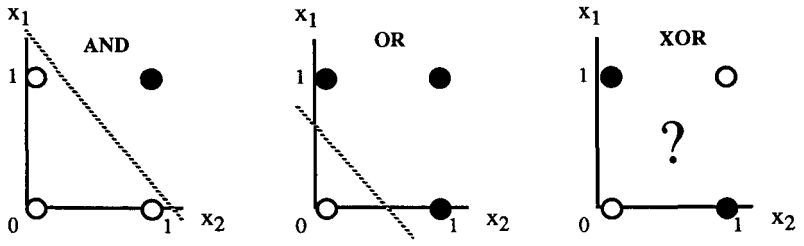


FIGURE 15. Geometrical representation of the Perceptron's input space

Four points exist: (0,0), (0,1), (1,0) and (1,1). It may be clear that in the XOR case, one cannot use a linear classifier to separate the open from the closed points. Therefore, the (single layer) perceptron cannot learn the XOR problem [Minsky et al., 1969/1988].

By introducing a hidden layer one can extend the perceptron so that, in principle it can make the classifications necessary for the XOR problem. However, in those days one did not know how to train a multi-layer perceptron, so this was not a relevant solution. In a multi-layer model, it is not known which neuron on which layer is responsible for a certain output value. Therefore, one does not known which weights to adapt. This problem is known as the *credit-assignment problem.*

One important issue was overlooked in this period. Perceptrons might not have been able to learn some mathematical functions in finite time, but they were very well able to model psychological behavior. For most researchers that should have been much more important than the message from Minsky and Papert. Nevertheless, this message did cast serious doubts on the viability of neural networks.

To some extent, the decline of the neural research school may have been due to Lord DARPA (Defense Advanced Research Project Agency) who decided not to spend any more money on a research direction that could not even solve the problem of exclusive OR. Much more was expected from the other sister of Machine Intelligence: the symbolic Artificial Intelligence community. Due to lack of funding, the neural paradigm went underground for more than a decade.

4.7 The Dark Ages

Researchers like Stephen Grossberg, Leon Cooper (a Nobel laureate in superconductivity), Christopher von der Malsberg, Shun-ichi Amari, Kunihiko Fukushima, David Marr, Teuvo Kohonen and James Anderson continued working on neural networks during the 70s. The last two are particularly well known for their simultaneous independent publications on associative memory models in 1972. These memory models used linear neurons and a Hebb-like training rule [Anderson, 1972][Kohonen, 1972, 1977, 1984].

Stephen Grossberg produced an enormous amount of research papers and several books from the beginning of the 70s up to now (most of which were very mathematical and difficult to understand on their own). Only recently did he start to simulate his models. Together with his group (and his wife, Gail Carpenter, in particular), he developed the ART and ART-2 implementations and used them in various applications. ART-2 was the first patented neural network algorithm.

By the beginning of the 80s, there were a number of different influences that triggered the revival of the learning machine. First, James McClelland and David Rumelhart developed a psychologically plausible model of character and word recognition. Although their model did not implement any learning (the weights of the connections had to be set by hand), it showed that characters in words were better and faster recognized than the same characters in non-words by implementing an efficient "Interactive Activation" model [McClelland et al., 1981][Rumelhart et al., 1982].

4.8 The Renaissance

In 1981 Richard Sutton and Andrew Barto analyzed the Delta Rule in much detail, providing the community with a much better understanding of this basic mechanism [Sutton et al., 1981].

However, the modern era of neural networks really starts with the publication of John Hopfield's 1982 paper on "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". John Hopfield was known as a distinguished physicist. So, his decision to study neural networks was very important for the field to be taken seriously. In fact, by relating the training of a network to seeking minima in energy landscapes he laid a theoretical foundation for neural research [Hopfield, 1982].

From that moment, it all seemed to happen simultaneously. Some important milestones achieved in those days were:

- Jerome Feldman and Dana Ballard from Rochester University, who made the term Connectionism popular and pointed out the biological implausibility of many of the Artificial Intelligence approaches in [Feldman et al., 1982].
- David Ackley, Geoffrey Hinton and Terrence Sejnowski develop the "Boltzmann Machine", a neural model similar to Hopfield's model but extended with a stochastic element. Moreover, they introduced "stimulated annealing", a technique to prevent a model from getting stuck in local minima.

4.9 Back-propagation

The real boom came after the 1986 publication of David Rumelhart, Geoffrey Hinton and Ronald Williams "Learning Internal Representations by Error Propagation." By showing the world how to train a multi-layer perceptron model with non-linear neural activation functions, a whole new research field appeared. Just as Teuvo Kohonen and James Anderson simultaneously published their work on associative memories, so was the back-propagation algorithm invented by different people at the same time. The credit for the algorithm goes to [Rumelhart et al., 1986a], who gave it the name "error back propagation" and related it to neural network research.

The two discoveries of the algorithm simultaneous to that of Rumelhart et al. were in the mid 80s were by Yan Le Cun and David Parker[1] [Le Cun, 1986][Parker, 1985]. However, already in 1974, Paul Werbos described an algorithm similar to back-propagation in his Harvard Ph.D. thesis [Werbos, 1974]. Some even go further back in history to seek the real roots of the back-propagation algorithm. Similar mathematical recursive control structures

1. Parker already anticipated the algorithm in 1982, but it took him to 1985 to actually work out the details.

can be found in the work of Arthur Bryson and Yu-Chi Ho in 1969 [Bryson et al., 1969]. It can even been shown that the structure of the learning algorithm follows from the so-called Robbins-Monro technique as introduced in 1951 [Robbins et al., 1951].

As mentioned, the back-propagation algorithm became the backbone of neural research. Every day, new applications appeared on the electronic news groups. The first IEEE neural network conference in 1987 in San Diego had more than 1,000 research papers. Particularly famous became the NETtalk implementation. So far, nobody had been able to develop a model that could learn how to pronounce English words. NETtalk could. Now, identical implementations are known for German and Dutch [Sejnowski et al., 1986][Dorffner, 88][Weijters, 90].

The back-propagation algorithm is a supervised learning algorithm. A back-propagation network consists of multiple layers: an input layer, one or more hidden layers (usually only one), and one output layer. By taking less neurons in the hidden layer than in the input and output layer, a distributed data representation in the hidden layers is forced, which yields the generalization and associational behavior of the model. The errors that are used by the learning rule for the adaptation of the weights of the hidden units are back-propagated from the errors found in the output units (see Figure 16, "The back-propagation network model," on page 48).

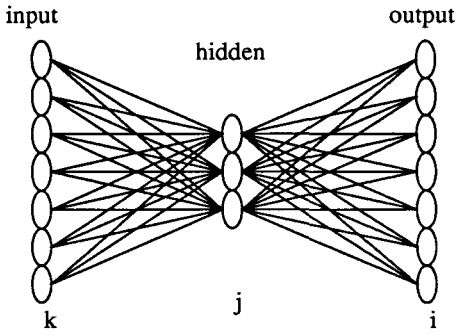


FIGURE 16. The back-propagation network model

If i, j and k represent the neurons in the output, hidden and input layer, then the activity s_j of neuron j of the hidden layer is given by:

$$s_j = \sigma\left(\sum_k w_{kj} s_k\right) \quad \text{(EQ 14)}$$

where σ is the so-called sigmoid function. See "Activation Functions" on page 34. The activity functions of the output layer are:

$$s_i = \sigma\left(\sum_j w_{ji} s_j\right) \quad \text{(EQ 15)}$$

The output error for the ν^{th} pattern in the i^{th} output neuron can be defined as:

$$\varepsilon_i^\nu = y_i^\nu - s_i(x^\nu) \quad , \quad \text{(EQ 16)}$$

A sigmoid function is used for which holds:

$$\sigma'(x) = \sigma(x)\,(1 - \sigma(x)) \quad \text{(EQ 17)}$$

Then the following weight update rules for the back-propagation rule can be given:

$$\Delta w_{ij} = \alpha \cdot \varepsilon^\nu{}_i \cdot s_j s_i (1 - s_i) \quad \text{(EQ 18)}$$

$$\Delta w_{kj} = \alpha \cdot \sum_i \varepsilon_i^\nu \cdot s_k s_i (1 - s_i) \cdot w_{ji} \cdot s_j (1 - s_j) \quad \text{(EQ 19)}$$

where α is a small constant.

Although the back-propagation algorithm meant a great step forwards for neural networks as a computing paradigm, there are a number of basic problems.

- First, the function E is a very complex function of all the connection weights. This function has numerous local minima. The gradient descent always leads to the nearest minimum, which may be higher the any global minimum.
- Next, the back-propagation algorithm has a number of learning parameters, which are not given by the algorithm. The convergence of the back-propagation algorithm depends greatly on these initial values. So, wrong initial values can cause wrong weights and thus a wrong mapping. Determining the proper learning parameters is often based more on guessing than on good reason.
- Finally, it may occur that the network has a very high (or very low) activation value, resulting in an activation value (due to the sigmoid function) near one (or zero). Then the weight adaptation shall be minimal and as a result, the training process will come to a complete standstill [Ritter et al., 1992].

4.10 Associative Memories

One of the first written works on memory can be found in [Aristotle, 400 B.C.]. Aristotle described the basic properties of what is known today as an associative memory. "We do not have an explicit store of knowledge, we are knowledge" was one of the statements in the beginning of this chapter. Aristotle was one of the first who was aware of this. He described a memory system in which objects are stored within a certain *distance* of related objects given certain *features*. Today, we call such memories: *associative memories*.

In particular, Aristotle defined the basic element of memory as a sense image. In addition, he defined associations as links between these memories. These links then served as the basis for higher-level cognition. Although Aristotle was quite unaware of the exact functionality of the memory elements, he tried to solve some typical memory problems. For instance the known problem of different versions between sense images in time was solved by tagging the various versions with temporal information. Finally, he proposed a method to compute with these images by using the links. Different types of links can cause association: temporal links, "something similar" links, "opposite" links, "neighboring" links. Aristotle defined this association process as a highly dynamical and flexible process, even as a kind of searching.

Associative memories have many important properties such as fault-tolerant processing, implicit generalization, and automatic error-correction. But, what is meant by distance and what are the exact features to base the distance on? Moreover, how are these features derived, expressed or normalized? And, how can the feature values of these associative memories be determined automatically?

Models such as the perceptron and back-propagation are interesting (supervised) pattern classifiers. They can be trained to represent (non-linear) function mappings. But they are not capable of implementing an associative memory in which related object are in neighboring areas. However, Kohonen feature maps, ART and some of the other self-organizing models can implement such a process.

In cognitive processes (of which language is one), associations and generalizations can be modelled much easier by using such an associative mechanism. One just has to check the direct neighborhood of a particular concept in order to find related concepts. Therefore the really interesting aspects of (high-level) cognitive behavior can probably be modelled better with self-organizing models instead of with function approximators such as back-propagation.

4.11 Kohonen's Self-Organizing Feature Maps

During all this back-propagation upheaval, Teuvo Kohonen continued working on his associative memories and his self-organizing feature map. Unlike back-propagation which was a supervised learning algorithm, the Kohonen learning rule does not involve supervision. Thus it can discover clusters of dependencies in unstructured data sets whereas the data input for back-propagation algorithm needed to be structured.

The Kohonen network is known to implement a vector quantization algorithm, well suited for clustering purposes. Originally, this model was an abstraction from the visual model presented by Von der Malsberg in 1973. Kohonen eliminated the interneuronal weight modifications during the training phase and simplified the adaptation of the weights. The feature map is an abstraction of the biological topology preserving maps found in the human visual system [Kohonen, 1982a-c, 1984, 1988, 1990a-b] [Malsberg, 1973].

Kohonen defines a one layer map, where all neurons are connected to the same set of input fibres. After determining the best match for an input vector, the weights of the input fibres (or synapses) of neurons within one region are changed according to the input excitation. By defining a horizontal inter-neuronal structure, lateral inhibition (used to determine the best match) is implemented. In certain cases, the system self-organizes, whereby regions on the map are formed, representing data falling within a certain statistical cluster, and thus automatically forming categories (dependent on the internal coding of the input patterns). By doing so, a highly efficient, powerful statistical classifier is obtained.

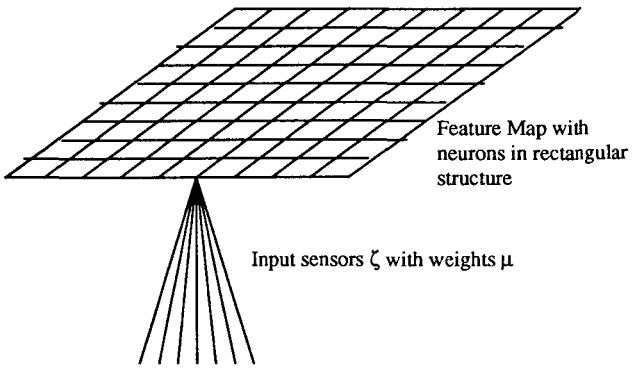


FIGURE 17. The Kohonen feature map

Kohonen feature maps implement a number of important properties:

- First, the feature map conserves the *topology* (that is, if two object are close to each other in the probability space, they will also be close to each other after the mapping) of the original probability distribution function of the data set, even after a dimension reduction.
- Next, feature maps implement an *automatic feature selection*. It does not matter whether one adds multiple redundant input sensors, the data is only organized on the relevant features given the context in which they occur.
- The feature map organizes on frequency as well as on context. That is, the frequency of a certain input pattern is just as important as the overlap between parts of these patterns. By doing so, feature maps implement a map of *conditional probabilities*.

The Kohonen formalism is a competitive learning algorithm. A two-dimensional map is constructed in a rectangular or hexagonal structure of individual neurons. Each neuron i has a number of input sensors with an input activation ξ_j and an input weight μ_{ij}. All neurons have the same number of input sensors and input weights. The activation of neuron i is calculated by:

$$y = \sum_{j=1}^{n} \mu_{ij} \cdot \xi_j \qquad \text{(EQ 20)}$$

Actually, this is not an activation function as we know it from the feed-forward networks. Here, the activation value is not used to feed-forward an activation value to the input of a

following layer, but it is used to determine a measure of correlation between the input activation and the weight vector of a neuron. The neuron (or area) best representing the input activation can be determined by finding the neuron with the highest output activity. This might be considered as the neuron r with the best match for input vector x for all neurons, or the minimum Euclidean distance between the vectors x and w[1].

$$\|x - w_r\| = min_i \|x - w_i\| \qquad \text{(EQ 21)}$$

The learning rule operates in the following way:

First, copy the activation values of an input vector x into all input activation sensors of all neurons. Next, determine the best match by finding the neuron with the minimum mathematical distance between input and weight values (this neuron can be referred to as *Best Matching Unit* or *BMU*). Then, adapt the weights of the neurons within a certain region of this minimum, so they'll recognize the current input vector better in the future. A general learning function is [Kohonen, 1984]:

$$\frac{d}{dt} u_{ij} = \alpha(t) \{ \eta_i(t) \xi_j(t) - \gamma[\eta_i(t)] \mu_{ij}(t) \} \qquad \text{(EQ 22)}$$

where $\alpha(t)$ implements a decreasing function in time in order to guarantee convergence, $\eta(t)$ is a function of the distance of the neuron to the BMU, and γ represents some non-linear scalar function of $\eta(t)$. This rule adopts the weights of the neurons in the neighborhood of the BMU.

By doing so, the BMU and the neighboring neurons all represent the input vector better after the weight update. So, it will be recognized earlier in future situations. Because the area surrounding the BMU is also updated, very interesting (recurrent) neighborhood effect occur during the training process, which enable the model to derive a topological map by self-organizing means. Without the neighborhood effects, the model would implement a much less sophisticated mapping.

If neurons are updated within a certain area of the BMU c, and one takes $\eta_i(t) = 1$ inside the area and $\eta_i(t) = 0$ outside the area, $\gamma(0) = 0$, and $\gamma(1) = 1$, then this equation can be rewritten as:

1. ξ and μ indicate *sensor values* of input values and weights. x and w indicate *vectors* of input values and weights.

$$\frac{d}{dt}u_{ij} = \alpha(t)\ \{\xi_j(t) - \mu_{ij}(t)\} \qquad \text{(EQ 23)}$$

Research carried out by [Ritter et al., 1989a] showed that "bell-shaped" functions perform best for $\alpha(t)$.

Rewriting these equations in vector format with w indicating the weight vector and x the input vector results in the following formulation of the learning algorithm:

- Initialize all weight vectors w with random values between 0 and 1.
- Iteratively consider different inputs, and determine the neuron s for which the best match between the input values x_s and weight values w_s among all neurons r in the feature map:

$$\forall r\ (\| w_s(t) - x(t) \| \leq \| w_r(t) - x(t) \|) \qquad \text{(EQ 24)}$$

- Update all weights according to the Kohonen Learning rule:

$$w_r(t+1) = w_r(t) + \varepsilon(t) \cdot \Phi_{rs}(t) \cdot (x(t) - w_r(t)) \qquad \text{(EQ 25)}$$

where:

$$\Phi_{rs} = e^{-\frac{\| r-s \|}{(2\sigma(t))^2}} \qquad \text{(EQ 26)}$$

$\| r - s \|$ is the physical distance between neuron r and s on the map, and

$$\varepsilon(t) = \varepsilon_{max} \cdot \left(\frac{\varepsilon_{min}}{\varepsilon_{max}}\right)^{\frac{t}{t_{max}}} \qquad \text{(EQ 27)}$$

$$\sigma(t) = \sigma_{max} \cdot \left(\frac{\sigma_{min}}{\sigma_{max}}\right)^{\frac{t}{t_{max}}} \qquad \text{(EQ 28)}$$

$$\varepsilon_{min} \in [0,1], \varepsilon_{max} \in [0,1], \sigma_{min} \in [0,1], \sigma_{max} = \frac{\sqrt{N}}{2} \quad \text{(EQ 29)}$$

Equation 34 is derived from equation 31 by substituting two bell shaped functions, Φ and $\varepsilon(t)$ for $\alpha(t)$. After numerous cycles, a topological map will be formed, holding related elements in neighboring regions.

A self-organizing process in time is given on the next page (see figure 19). Here, a two-dimensional feature map with two-dimensional sensors (ζ_1,ζ_2) is used to map a homogeneous distributed set of input vectors from $\mathbf{R}^2$ to $\mathbf{R}^2$ (similar plots can be made for mappings with different dimensions).

The neurons can obtain all possible values in the domain [0,1]. However, the two dimensions in the training vectors can only obtain values from the set:

$$\{\frac{1}{7}, \frac{2}{7}, \frac{3}{7}, \frac{4}{7}, \frac{5}{7}, \frac{6}{7}, \frac{7}{7}\} \quad \text{(EQ 30)}$$

The training set contains one element of all possible vectors in this domain. That is, the training set holds the following vectors:

$$\begin{pmatrix}\frac{1}{7}\\ \frac{1}{7}\end{pmatrix}\begin{pmatrix}\frac{1}{7}\\ \frac{2}{7}\end{pmatrix}\begin{pmatrix}\frac{1}{7}\\ \frac{3}{7}\end{pmatrix}\begin{pmatrix}\frac{1}{7}\\ \frac{4}{7}\end{pmatrix}\begin{pmatrix}\frac{1}{7}\\ \frac{5}{7}\end{pmatrix}\cdots\begin{pmatrix}\frac{7}{7}\\ \frac{7}{7}\end{pmatrix}$$

During the training session, vectors are selected randomly from this set. every factor has an equal probability to be selected. Therefore, the probability distribution of the input space is considered homogeneous.

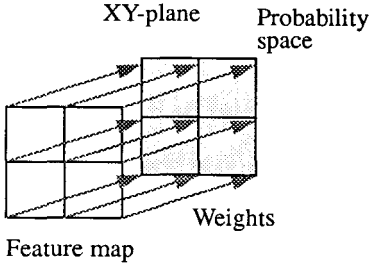


FIGURE 18. In the self-organizing state, the weight vectors of the feature map represent specific points in the probability space.

By plotting the values of all weight vectors in a XY-plane, and by connecting each neuron with its direct neighbors, a perfect rectangular graph is obtained if and only if the feature map reaches the so-called self-organizing state. In this state, the feature map is perfectly ordered, that is: all neurons represent data that have a minimal Euclidean distance to all its neighbors. In that case, only one solution is possible, the graph forms the same map as the original topology of the feature map, in this case a perfect rectangular shape.

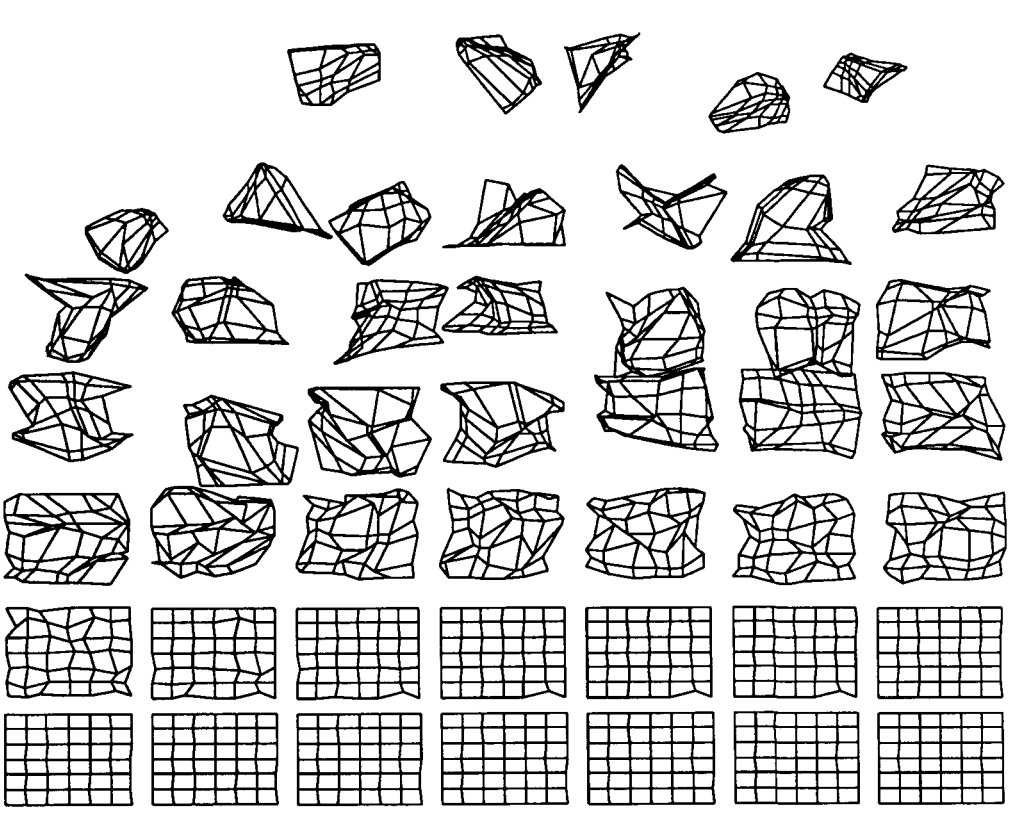


FIGURE 19. The Self-Organizing Process. From random weights (upper-left corner) to the self-organizing state (bottom-right corner).

4.12 The Hypermap

The Hypermap is a more context-sensitive variation on the feature map. It was introduced by [Kohonen, 1991]. The normal feature map is not particularly context sensitive, making it hard to use in some cases. In the hypermap training algorithm, the data is presented within its (natural) context as a concatenated vector of the object data x_o and the context data x_c.

$$x'(t) = [x_o(t), x_c(t)] \quad \text{(EQ 31)}$$

$$w'(t) = [w_o(t), w_c(t)] \quad \text{(EQ 32)}$$

The context can be a shifting window as well as a more artificial notion of context (like a context category). First, the feature map is trained on this context only by using the regular training rule. During this first training phase, the object weight values stay fixed.

$$w_{c_r}(t+1) = w_{c_r}(t) + \varepsilon(t) \cdot \Phi_{rs}(t) \cdot (x_c(t) - w_{c_r}(t)) \quad \text{(EQ 33)}$$

After a certain saturation value is reached, the context weights are fixed and the map is trained on the object sensor values.

$$w_{o_r}(t+1) = w_{o_r}(t) + \varepsilon(t) \cdot \Phi_{rs}(t) \cdot (x_o(t) - w_{o_r}(t)) \quad \text{(EQ 34)}$$

By doing so, the global properties of the data distribution are captured first, while the details are determined later.

It appeared that these maps showed a 10-20% performance increase in speech recognition applications. Moreover, this adapted training algorithm performs much better in large feature maps (over 1000 neurons) because they get infolded or tangled less easily.

4.13 Neuronal Group Selection and Genetic Algorithms

Even more biologically inspired than Linsker and Von der Malsberg, is the work of [Edelman, 1987] and [Reeke et al., 1988]. Their Neuronal Group Selection (NGS) theory applies a genetic approach to the formation of maps on the cortex and the growth or change of neural connections, which constitute the learning processes of human beings. Their models have learning rules that model populations of organisms as populations of distributed patterns within neuronal groups. Each neuronal group represents a solution to

the problem. Better solutions (e.g. groups) multiply (or grow) faster than worse solutions, resulting in a good (or optimal) solution after a few generations.

This biologically and evolutionary inspired theory states that human genes do not have enough memory capacity to encode the structure of the human nervous system before it actually develops. This, together with the fact that even twins have completely different nervous systems, implies that there has to be some way of "competitive growing" or, in other words, some kind of Darwinist process that eliminates structures that are irrelevant or unsuccessful (according to some fitness function).

Edelman and Reeke criticize connectionism for avoiding the selectionist aspect in its current models. Their strong point is the clear absence of identical neuronal structures in nature: even twins exhibit different neuronal structures. One can compare this with the work done by [Goldberg, 1989] on genetic algorithms, which are much in use for the optimization of back-propagating neural networks.

4.14 Hybrid Models

Another variation seen in different research work is the level of connectionism. Connectionist solutions may not always be best as stated in an earlier paragraph. One can decide to use symbolic methods to solve sub-problems for various reasons, like complexity and performance. Accordingly, the literature provides us with a number of hybrid solutions in neurolinguistics.

Modules may be replaced by symbolic methods, e.g.: a lexicon for automatic feature detection and concept representation in [Miikulainen et al., 1988a-b]. Another solution to avoid the pre-wiring and pre-categorizing disadvantage of back-propagation can be found in [Hendler, 1989]. The addition of a conventional parser as preprocessor is used in [Waltz et al. 1984, 1985]. Even more obvious are the implementations of symbolic methods in connectionist systems. [Touretzky, 1987] tackles the slot filler problem by using a connectionist knowledge representation scheme. [Dolan et al., 1987] proposes a scheme for implementing schemata in a connectionist network. [Touretzky, 1986] implements BoltzCONS (a combination of a Boltzmann Machine and the basic list construction operation CONS in LISP), a recursive mechanism resulting in a connectionist production system [Touretzky et al., 1988]. Even Connectionist expert systems can be found [Saito et al., 1988], [Gallant, 1988], [Bounds, 1989], [Bradshaw et al., 1989], and [Gutknecht et al., 1990].

This list becomes longer every day, mainly due to the infinitely possible combinations of many conventional symbolic AI and new connectionist techniques.

4.15 The State of the Art

From the summer of 1992, there will be a split within the neural network research community. On the one hand, the more biologically inspired research group will start organizing its own conferences in computational neuroscience and behavioral sciences. On the other hand, there is a growing group of people who are trying to relate traditional statistical pattern recognition and neural networks. This second group is more interested in abstract models such as the back-propagation model and Kohonen feature maps.

At this very moment in 1992 neural network research is still flourishing, although it has lost some of its magic over the years as the working of the algorithms became more clear and less exciting. Currently there are two International Joint Conferences on Neural Networks (IJCNN) every year (one in the U.S. and one in the Far-East). Each of them still attracts more than 2,000 participants. Europe has its own large conference in the form of the International Conference on Artificial Neural Networks (ICANN) which was about the same size as each of the IJCNN's. In addition, large international Artificial Intelligence conferences such as the IJCAI[1], ECAI[2], SPIE[3], COGSCI[4], ICML[5], and AAAI[6] have important neural network sessions. It seems, neural networks have established a much stronger position in the research community than they had in the 60s. Therefore, this time they will probably remain, at least for their unrivaled power in applications such as handwritten Optical Character Recognition and the prediction of non-linear time series such as sun spots and stock market prices.

1. Biannual International Joint Conference on Artificial Intelligence
2. Annual European Conference on Artificial Intelligence
3. Annual Conference of American Society of Physics
4. Annual Conference of the Cognitive Science Society
5. Annual International Conference on Machine Learning
6. Annual Conference of the American Association of Artificial Intelligence

5.0 Radical Connectionism and the Psychological Plausibility of Artificial Neural Networks

Artificial neural networks are much less sophisticated than the human nervous system. However, a number of basic features can be defined that provide the specific properties of neural networks as they are referred to by many researchers these days.

If a neural network has the following properties:

- Massive parallelism,
- Natural data input,
- Distributed data representation, and
- Self-organizing

> *Only then* can one refer to the advantages of biological neural networks such as adaptive behavior, implicit generalization, error-correcting capabilities, and fault-tolerant processing.
>
> *If* on the other hand, one of these four features is missing, one should be extremely careful referring to the typical characteristics of biological neural networks.

Mainly responsible for the generalizing capabilities is the distributed data representation. However, if the representation does not use a massively parallel network, it will not be good enough to provide generalization in all cases. If a neural network does not implement any form of generalization it will be equal to a lookup table and therefore not worth the classification *neural network*. The same holds for the fault-tolerant processing and error-correcting capabilities of neural networks, which is in a way nothing else than generalizations into the correct data sets.

The adaptive behavior of a neural network is mainly due to a (self-organizing) learning rule that generalizes over known cases and that classifies new cases in known and new categories. If one does not use natural data input, a (manual) preclassification is carried out in which important information gets lost. One of the main properties of self-organization is that it performs automatic feature selection, given the context in which an object occurs. By artificially labeling an object, it can only be classified or organized on the basis of this artificial labelling. Therefore natural data input and self-organization are essential properties of artificial neural networks. In [Dorffner, 1991] the author mentions the four points above as properties of "radical connectionism", opposing it to the more moderated forms of connectionism. However, here, it is rather seen as "essential" in every connectionist modelling project that implies typical neural behavior or pretends to be psychologically plausibility.

6.0 Expected Problems

On the one hand, neural networks seem to be able to outperform symbolic methods in various ways. On the other hand, many problems in practical neural network use remain unsolved. This section will discuss some disadvantages of neural networks.

6.1 Precise Computations, Dynamic Binding & Hierarchical Structures

First, neural networks are poor at *precise computations*. Therefore, sequential computers will outperform neural networks in these computations. This shortcoming of neural networks might not be a problem in natural-language processing, because precise computations may not be needed there.

Second, *symbolic reference*, *dynamic binding* and *hierarchical structures* are essential elements of all cognitive theories. Therefore, there should at least be a functional equivalent for these mechanisms in neural networks. However, the most significant problems for neural research to solve are the dynamic binding problem and the mapping of hierarchical structures to vectors.

The first problem is a direct result from the fact that distributed neural networks are relatively unstable memory elements for exact data. Therefore one cannot store a variable in a particular place for an unlimited time. The second problem is a special case of the first problem. Due to the instable memory elements, recursive and hierarchical structures cannot be stored for an unlimited time.

Natural language is one of the problems in which structure is explicitly present by means of syntactical structures. However, neural networks are only capable of processing vector elements. So, how does one map these hierarchical structures to vectors without loss of information? Even more important, how does one map the vectors back to a hierarchical structure?

Both questions are not solved yet. However, once they are, the sky will be the limit for the application of neural networks in natural language processing as well as other problems involving some form of learning and processing of hierarchical structures.

6.2 Temporal Sequences

In a sequential computing paradigm, sequences are implemented implicitly by the data flow. In parallel systems however, one has to implement a special mechanism to keep track of sequential dependencies. Neural networks do not appear to be good in representing changes in time. It is quite easy to develop a neural network that realizes some sort of classification function, but the implementation of time dependency is another story. There are a number of time influences in decision making: time- varying responses (long term by changing the weights of the interconnections; short term to represent time in a network) and sequence (analyzing inherently sequential input). Overall, three solutions can be given to the short-term timing problems and the sequence handling. Most simple is the addition of extra input bits, which represent the timing information explicitly. This solution does not have any biological plausibility and it results in network patterns that are too complex.

A second method is the construction of extra layers in the network with memory functions: so called feedback (or recurrent) loops, in particular popular in back-propagation [Jordan, 1986] [Elman, 1988]. In principle, all self-organizing models implement a form of recurrence by adopting their connection weights through local interaction with their neighbors only. Addition of other forms of recurrence quickly increases the complexity and may result in unstable models.

Another solution is a shifting window structure. Instead of using feedback connections to implement sequences, the data is presented in parallel to the neural network. By presenting the data within its natural context, the system becomes aware of sequences. The shifting window prevents the model from only processing fixed (or maximum) length data input. However, the model will never be able to develop an internal memory structure that represents data dependencies longer than the window size. Moreover, the model also implements a finite state machine [Ritter et al., 1990], [Sejnowski et al., 1986].

Already in the late 50s, a famous paper on the psychological plausibility of shifting windows was written by [Miller, 1956]. One of the main problems in interpreting this work is the question of what is shifted over: characters, words, sentences, or mental concepts?

6.3 General Criticism on Connectionist NLP by Neuropsychologists

Various researchers in traditional artificial intelligence have plenty of criticism on the connectionist language processing paradigm, but then, it is exactly they who are attacked by the PDP group. In their turn, biologically oriented neurologists propose serious criticism on the neuro-psychological plausibility of some aspects of the PDP models [Gigley, 1983, 1985].

First, PDP models include binary feature detectors and a mutual inhibition scheme to recognize input. From a neuropsychological viewpoint, this assumption is wrong. Only the recognized input is active, not the opposites or synonyms. According to Gigley, there is no inhibition of any information at the perceptual level. Therefore explicitly defining mutual inhibition is wrong.

Besides the psychological problems of lateral inhibition, there is also a practical one; all the inhibitory interconnections must be prewired by hand, an enormous amount of work.

This non-adaptive character of the back-propagating algorithm and the need to define all the micro features by hand, might convince the reader that self-organizing techniques are better in natural language processing. Critique from Edelman and Reeke also points in that direction [Reeke et al., 1988].

Although this critique definitely applies to all artificial neural networks, one should be aware of the fact that NLP problems are complex by their very nature. Therefore, one can defend the position that one must be extremely careful in implementing the already complex NLP models in even more complex neural models.

6.4 General Criticism on Connectionist NLP by Cognitive Scientists

[Fodor et al., 1988] advocate the position that neural networks are never stable enough to implement a dynamic binding system between variables (or symbols) and values. Without such a system, information on certain topics should be present in exactly the same form at different positions in the brain at exactly the same time. As this can never be the case, some form of symbolic reference process must be present in the human brain. As long as current artificial neural networks cannot model such a process, they do not model the human brain properly and should therefore not claim to be more psychologically plausible than current symbolic Artificial Intelligence techniques.

In other words, one needs to implement:

- *Compositionality*; the recursive combination of symbol structures in larger structures, and
- *Distal access*; a mechanism to refer a remote structure through some pointing device (needed by the compositionality).

Current neural networks are not able to implement (recursive) hierarchical structure without the loss of typical neural properties such as distributed data representation and automatic learning.

Chapter 2

Neural Networks in Natural Language Processing and Information Retrieval

"Every time I fire a linguist, my performance goes up"
-- Frederick Jelinek

Abstract

Over the last decades, many applications have been realized by applying neural network technology toward natural language processing and information retrieval. In this chapter these efforts shall be discussed in detail.

In order to present the models in an understandable manner, they are categorized according to the data representation scheme use: local, sub-symbolic or distributed. At the end of this chapter, a number of important objectives that have to be met in order to successfully apply neural networks to natural language processing and information retrieval shall be given.

1.0 Background

Connectionism is often seen as the paradigmatic competitor of the symbolic processing tradition in artificial intelligence [Graubard, 1988]. Recent renewed interest in the field is mainly caused by the limitations of the symbolic methods and the practical problems occurring in the implementations of parallel algorithms.

Lately, explorations in natural-language processing (NLP) and neurocomputing were combined in the new field of connectionist NLP. In particular the property of connectionist systems to distribute knowledge, with conservation of generality and integration is interesting for NLP research.

A proper functioning natural-language understanding system should combine knowledge from many different sources like syntax, semantics and pragmatics. The interaction between these knowledge bases seems to require a very broad bandwidth. This was not anticipated by the architecture of NLP systems. Many older NLP systems suffer from problems indirectly caused by these limitations. In [Charniak, 1983], marker passing components are introduced, which run in parallel with the syntactical and the semantical analysis and keep track of common interests. Other attempts to integrate different sources of knowledge can be found in the study of blackboard systems [Hayes-Roth, 1985], and in the symbolic integration of syntax and semantics. All of these systems were unable to integrate the different knowledge sources effectively.

Another argument applies in the field of Information Retrieval (IR). Here, only statistical pattern recognition techniques have been used. In general, the level of analysis does not exceed that of a simple pattern matcher. Language issues such as structure and meaning are ignored completely. There is no integration whatsoever of different knowledge sources. Only statistics are used to implement simple (mostly adjacent) context dependencies.

The connectionist approach offers a massively- parallel, highly-distributed and highly-interconnected solution for the integration of various kinds of knowledge, with preservation of generality. It might be that connectionism or neural networks (despite all currently unsolved questions concerning learning, stability, recursion, firing rules, network architecture etc.), will contribute to the research in natural-language processing and information retrieval.

Parts of the work presented in this chapter have been published in [Scholtes, 1990a-b].

2.0 Introduction

Connectionist NLP might be set forth as the combined study of computational linguistics and artificial neural networks. However, both these research fields lack a good definition themselves. On the one hand, the linguistic part has various definitions of language and is involved in many different types of language processing. Translation, understanding, generation, interfacing, speech recognition and optical-character recognition (OCR) are among the various applications of NLP.

On the other hand, neural-network research involves many more aspects than the name by itself suggests. Different aspects of neural networks and computational linguistics might be covered by the term neurolinguistics. Therefore there is a need to limit the scope of the term in this context.

Computational linguistics (CL) encompasses both the application of a computational paradigm to the study of human language and the design and implementation of systems that process or analyse written or spoken language. The also widely used term natural-language processing (NLP) in particular focuses on the second part of the definition: the engineering.

The linguistic theories used in NLP were more often than not derived from a different context. A good example is generative grammar [Chomsky, 1957, 1965]. The aim of linguistic models is the formalization of language. They are certainly not meant to be used within a NLP system.

The unintended use of linguistic theories by computer engineers in NLP systems is one of the reasons for the problems in NLP. Linguistic models have a completely different nature than models suited for computer implementations.

Through all levels and directions of linguistic research, i.e. context, syntax, semantics, pragmatics, understanding, perception, generation, parsing, querying, etc., the problems occurring nowadays are best characterized by the following key words: complexity, ambiguity, robustness, maintenance, learning and generalization.

3.0 Neural Networks in Natural Language Processing

Connectionist NLP encompasses many combinations of the previously mentioned directions in neural network research and computational linguistics. For example: speech recognition, speech generation, optical character recognition (OCR), parsing, perception and disambiguation. Because neural networks are good at pattern recognition, they are mainly used in speech analysis and OCR. However, here the focus will be on parsing, perception and disambiguation.

In chapter one, neural networks were classified by using the dimensions *neuron, training rule* and *network topology.* However, as time passed by, another dimension could be distinguished: that of the *structure of the data representation* put on the input sensors. The first systems available tried many different types of neurons, training rules and network topologies, but they had one thing in common; they all used a local data representation. A second generation of connectionist NLP systems could be characterized by a sub-symbolic data representation, whereas the most recent models use a fully distributed representation scheme. In this section, these three types of applications are discussed in separated sub-sections.

In addition, a remark has to be made with respect to the internal encoding scheme of connectionist models in NLP in general. Many neural net models are involved in natural sensor processing (for example frequencies in speech recognition, light intensities in vision, and movements in robotics). In connectionist NLP, the sensor values are artificial, that is, they are assigned by some kind of lookup table: a symbolic object is translated into an artificial vector code.

3.1 Localist Systems

As mentioned, connectionist systems with a local data representation are in fact nothing more than a complicated implementation of symbolic information processing paradigms. The local system behaves as a parallel lookup table and has nothing to do with what (biological) neural networks stand for. Two of these *connectionist natural language processing systems* that could be observed in the beginning of the 80s are:

- The word-recognition model by David Rumelhart and James McClelland, and
- The language disambiguation models by Gary Cottrell and Steven Small.

None of these models could be trained automatically neither did they generalize or show error correcting behavior. They were mainly studied as parallel activation models [Feldman et al., 1982]. Most of the localist connectionist work in those days was carried out at the University of Rochester.

Localist systems were mainly evaluated for two effects that occurred: spreading activation and lateral inhibition.

In *spreading activation*, decisions are spread over time, so various knowledge sources can propose elements of interpretation. Two types can be distinguished: digital and analogue. Digital spreading activation can be found in [Charniak, 1983], where marker-passing algorithms run in parallel with the parsing and semantical processes, using a depth first search algorithm to select correct interpretations of objects. The analogue type involves a network of weighted associations where activation energy is spread over the network as a mathematical function of the strength of the interconnections [McClelland et al., 1981] [Rumelhart et al., 1982]. Both have an overkill effect. The digital form often results in too many search paths. Analogue spreading of activations can result in activation of the entire network. Instead of using techniques like damping and decay (carefully chosen weights between units, so not the entire network is activated), the model uses lateral inhibition, as proposed in [Feldman, 1981] for modelling biological-vision systems.

Lateral inhibition prevents two opposing action systems to excite simultaneously. In parsing this means nodes representing alternative analyses of the same input may not be activated at the same time. By connecting them with inhibitory connections, only one of them will survive when the network relaxes, so there will be no conflict. To be more explicit: the most likely one will survive. Nodes representing different lexical categories for the same word, nodes representing different senses of the same word, nodes representing conflicting case role assignments, and corresponding semantic or syntactic interpretations should be interconnected in a similar way.

3.1.1 A Word Recognition Model

One of the first simulations of content-addressable memory in language can be found in the model as developed by [McClelland et al., 1981]. The authors found that recognizing letters in words was much easier than in non-words. This effect was also found for pronounceable non-words compared to unpronounceable non-words.

The model consisted of three layers. First the feature input layer, which received input from two sources: visual and acoustic. These two detectors worked independently. Both units were connected to a higher level, containing a letter and phoneme unit. These two units are highly interconnected with the third word level. By exchanging information, the word level relaxes into a state representing the recognized word. The recognition of pronounceable non-words was probably caused by the familiarity of the network with the phonemes.

To describe the model's general characteristics in terms of connectionist models, it can be said that McClelland and Rumelhart developed a multi-layer network (features, letters,

words) with local-knowledge representation (one object on unit). No learning algorithm was developed; the local representation made it possible to set the connection strengths by hand. A distributed-knowledge representation scheme would need a powerful learning method. The units behaved like neurons. Relaxation of the network was achieved by competing neurons. The winning relaxation state of the network was usually the state representing the correct word. A more in depth study of the model can be found in [Rumelhart et al., 1982].

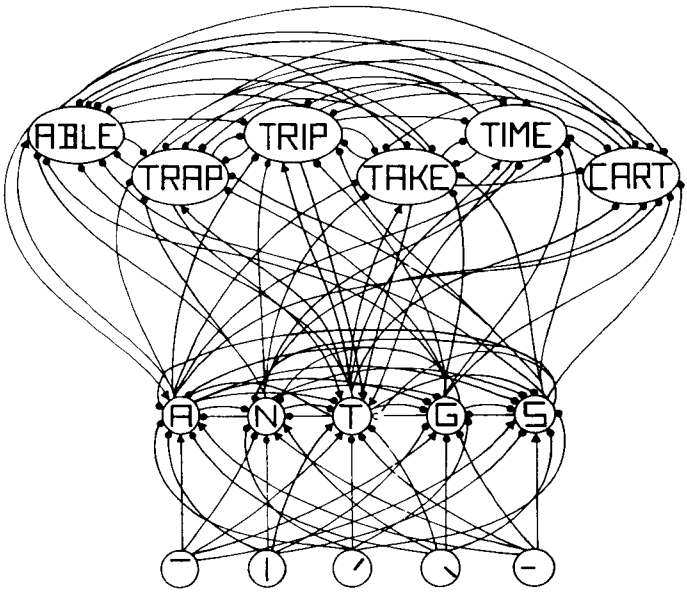


FIGURE 1. An interactive activation model of context effects in letter perception (reprinted from [Rumelhart et al., 1986c]).

3.1.2 A Language Comprehension Model

In [Small et al., 1982] the authors reveal the need for a more integrated solution to the language comprehension process. According to their point of view, the richness of the interaction between the different sub-processes is heavily underestimated. A highly-parallel, highly-distributed and highly-interconnected model is developed, which is psychologically as well as computationally acceptable. Although the authors claim to use a distributed-knowledge representation scheme, the model still uses one unit for one concept; therefore it looks more like a localist solution then like a distributed one. The model con-

tinues the work of Rumelhart and McClelland, discussed above. Input comes from a phoneme perception network as in [McClelland et al., 1981].

The model consists of three layers: lexical, word sense and case logic. The lexical and word-sense level have the same function as similar modules in symbolic NLP systems. The word sense layer can be divided into four interconnected parts: noun phrase, grammatical, morpheme, and predicate information. Because the model behaves according to localist-connectionist concepts, it has more characteristics of a fancy (highly-integrated and interconnected) blackboard implementation, than of a distributed-knowledge representation scheme.

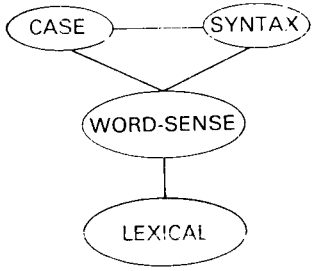


FIGURE 2. The system architecture (reprinted from [Cottrell, 1989]).

The main idea underlying this concept was the conviction that each word should be its own expert instead of integrating separate modules for different kinds of knowledge. The problems occurring in symbolic knowledge integration and knowledge maintenance are not quite solved in this model. Another disadvantage is the prewiring demand. All knowledge must be coded by hand in the interconnections. In [Cottrell et al., 1983], [Cottrell, 1985, 1989] the model is discussed in more detail. In particular, the authors spend much attention to connectionist solutions of the case role assignment problem [Fillmore, 1968].

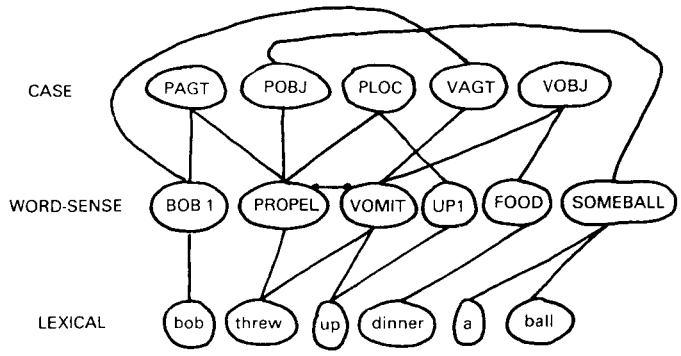


FIGURE 3. Network needed to disambiguate "John threw up dinner" and "John threw up a ball" (reprinted from [Cottrell, 1989]).

3.1.3 The TRACE Speech Recognition Model

Related to natural language processing is the problem of speech recognition. Much of the early (successful) neural network research was carried out in this field. In the TRACE model as introduced by McClelland and Elman, three levels of processing units can be distinguished. One for speech features, one for phonemes and one for words. Some of the features used are vocalization, diffuseness and acuteness. Time is represented by mapping it into space. The units represent quantized phonemes at a time slice.

The model combines information from all three levels to achieve proper word recognition. Therefore a number of interesting properties can be observed in the model. First, words are recognized earlier than non-words. Second, the model converges towards the most plausible words given a certain (phoneme) context at ambiguous input.

The name TRACE goes back to the property that all units are actively involved in the recognition, also units that were involved in phoneme recognition in earlier time slices. In fact, one can trace back the analysis of the speech input through the entire network.

Some major disadvantages of the model are the local data representation, the inability to determine the connection weights automatically and the enormous amount of processing units needed due to the time quantization [McClelland et al., 1986].

3.1.4 Constructing Connectionist Networks for Context-Free Grammars

Much effort has been invested in the automatic construction of localist neural networks that can be used to parse context-free grammars (CFG). In [Nijholt, 1990] and [Sikkel, 1990] the construction of such a network is compared to the determination of the non-terminal structures. By ignoring the terminals (the words in a sentence), a basic structure of a CFG can be found, which can than be used to represent the structure of the neural network:

> "The words become irrelevant, only the structure of possible sentences is important."

This method of constructing a (localist) neural network is denoted by the notion of *meta-parsing*.

3.2 The Subsymbolic Systems

Already in the early days of connectionist natural language processing, one was aware of the limitations of local data representation. However, a training algorithm for multi-layer, fully distributed systems was not known yet. As the localist systems resembled much to symbolic data processing, the subsymbolic models were somewhere in between of the symbolic models and the fully-distributed neural networks. The subsymbolic model has an intermediate level of structure between the neural and symbolic levels. The subsymbolic models all used hand-structured sets of (micro) features instead of flat (non-structured) vector input, which distinguished them from the fully-distributed models as discussed in one of the next sections.

The sub-symbolic paradigm was the next logical step from the study of symbolic systems to that of fully distributed processing. Much of the work done by the PDP group in the mid 80s could be defined as sub-symbolic information processing. It was the opinion of many that cognition is described at the sub-symbolic level, implemented by a connectionist model [Smolensky, 1987]. Their views were shared by other researchers such as Douglas Hofstadter.

In order to realize a coalition of nodes representing a consistent interpretation that will dominate after several interpretations, activation links are made between phrases and their constituents, words and different meanings, roles and fillers, and corresponding semantic and syntactic interpretations. Until the level of semantic interpretation, local representations are used.

By introducing microfeatures of meaning [Hinton, 1981], distributed-knowledge representation schemes are added to the model. Microfeatures have the powerful ability to define semantical concepts in terms of basic features that are shared with other concepts. So,

meaning is represented by a pattern of microfeatures, distributed over the network[1]. Although the model does not perform learning and the model is not completely integrated, the use of lateral inhibition and microfeatures provides a valuable addition to connectionist systems.

3.2.1 Massively Parallel Parsing

The first real distributed knowledge representation in sentence interpretation can be found in parts of the model developed by [Waltz et al., 1984] and [Waltz et al., 1985]. Their system, like several others discussed here, is not completely connectionistic, because they used a conventional chart parser to create parse trees that were used as input to their system. The main objective of the system was to resolve lexical ambiguity.

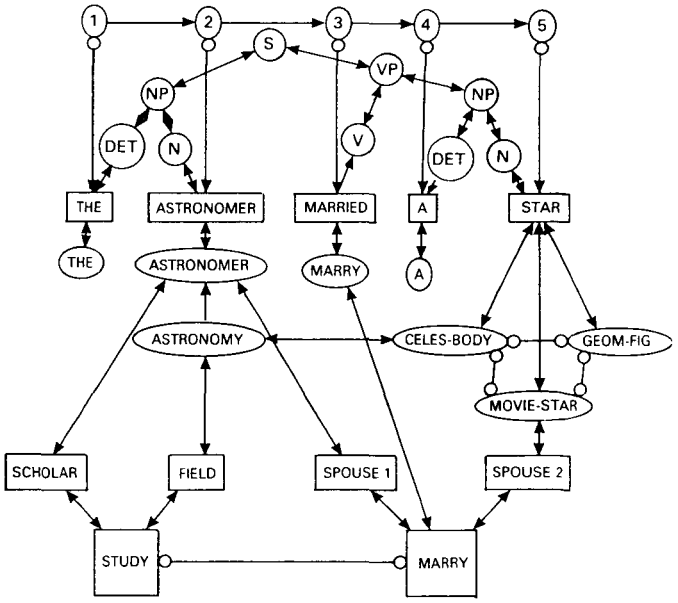


FIGURE 4. Network to disambiguate "The astronomer married a star" (reprinted from [Waltz et al., 1984]).

1. The terminology used in this context can be quite confusing. These models were called distributed because they used a distributed internal coding for the semantic part of the code. However, these early models still used a local representation for other parts of the code. All distributed representation schemes were man-made, and no internal distributed representation whatsoever could be derived automatically. Therefore one would rather classify these systems as sub-symbolic to distinguish them from the trainable fully-distributed models as presented in the next paragraph.

Although cooperative application of many knowledge sources e.g., word use, word order, phrase structure and real-world knowledge is acknowledged, these knowledge sources are seen as decomposable. While splitting the problem in different modules facilitates computation, a system which merely conjoins all these knowledge processing modules in a serial or hierarchical way, is not psychologically plausible. Therefore the authors choose for a highly integrated solution. It follows the work of [McClelland et al., 1981] and [Small et al., 1982]. The goal to develop a "model that runs like an ecological system" is achieved by combining spreading activation and lateral inhibition concepts.

In [Jodouin et al., 1990] the strengths and weaknesses of the Waltz & Pollack model of natural language processing are investigated. The authors try to extend the model, so it can be used in cases greater than single sentences context. In particular the relative fragility of the model was criticized. By extending the model with a more Hebbian learning rule, some of its weaknesses could be overcome. [Santos, 1989] tackles one of the basic restrictions of the Waltz & Pollack model: the constraint of fixed length sentences. By introducing a inter-neuronal shift mechanism, parse trees of variable lengths can be represented in the network.

A connectionist parser for arbitrary context-free grammars can be found in [Fanty, 1985]. Here, a hierarchical model is presented that parses maximum length strings. The most interesting property of this model is that it is purely syntactical, resulting in a model that generates all possible parses (most connectionist parsers just converge towards the most preferred pars given some syntactical and semantical features). The data representation is very local, yielding a high demand for processing units; all possible matches and productions must be represented by units.

3.2.2 Learning the Past Tense

A number of interesting issues in connectionist language processing are found in [Rumelhart et al., 1986b]: learning the past tense of English verbs and the role assignment of constituents. The first paper on past tense learning states that the internal representation of grammar rules is implicitly present in the human mind, but certainly not explicit in the form of explicit logical propositions, as proposed by some linguists. Therefore, a model is developed which uses a non-explicit knowledge representation form for comprehension and production of language.

By implementing such a model, the past-tense learning process of children is simulated. When children start talking, they only use frequently occurring verbs, which are almost all irregular. They just know the past tenses of these words, without referring to explicit grammar rules. The second phase involves learning grammar rules. At this moment, children start making errors with the irregular words (e.g., "goed"). In the third phase, chil-

dren regain the use of correct irregular forms of the past tense. This regularization phase continues until adulthood.

A simple one-layer pattern-associator network with a linear-threshold function [Kohonen, 1984] is used to implement the model. By giving the network probabilistic behavior (in fact the units fire probabilistically), learning speed is delayed and the system responses differ in time with the same input and the same network. The connection strengths are learned with the perceptron convergence procedure (discrete variant of the delta-rule). Linear feed-forward networks have the disadvantage of learning no more than N linear independent input vectors with N the number of input-bits. Therefore memory capacity is limited. The words are represented by Wickelphones (a kind of trigrams or third order Markov chains). The features (i.e. vowels, place in Wickelphone, etc.) are represented as a distributed pattern of activation over the set of feature detectors. Although the model was oversimplified, it did simulate the learning process of past tenses.

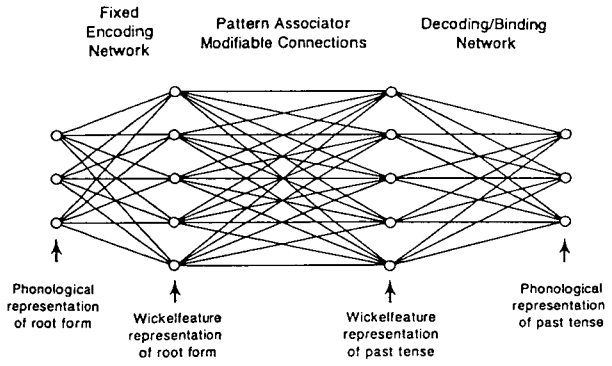


FIGURE 5. The architecture of the model (reprinted from [Rumelhart et al., 1986b]).

[Pinker et al., 1988] criticized Rumelhart & McClelland's model. First, the wickelphone representation was found to be insufficiently powerful to model all possible phonetic transformations that are found in natural languages. Second, many linguistic transformations, such as morphological ones, occur in many contexts. By issuing only one rule for one transformation, one does not allow an independent rule system to be generalized across a series of different contexts.

Much of this criticism is mainly caused by the limited nature of Rumelhart & McClelland's model. A larger scale system could handle different inflections by different rules if parsing, semantics and morphology were handled in parallel. In [Pinker, 1989] the author reveals that despite the comments and shortcomings of the model, it was the first one to show some plausible characteristics of language acquisition, a statement former models like [Harris, 1974] and [Anderson, 1977] did not dare to make.

3.2.3 Word Repetition

Another early model of human language processing, was mainly motivated by the wish to explain the word repetition effect: words seen before are recognized faster than others. [Rueckl, 1986] presents a three level model. The first layer selects features from a visual input. The second one projects these features in so called object-centered coordinates. Finally, the third level classifies or recognizes the objects. Moreover, humans are capable to use the same symbol for different visual effects (i.e. lower and upper case characters). Therefore a fourth module exists, holding internal representations for the visual objects. recognition of objects is done by relaxation, causing frequently seen objects to relax faster than in-frequent ones.

3.3 Fully Distributed Systems

The first fully distributed connectionist language-processing models could be observed at about the same time as the invention of back-propagation. Here, the entire coding is distributed (n units, m concepts). The signals that are presented to the input of the model is no longer structured by hand as is the case in the sub-symbolic paradigm. By using such coding schemes, generalization, association and robust behavior become implicit features of the models.

Here, it is claimed that these fully-distributed models are indeed different in more than one sense from the symbolic as well as the statistical NLP models.

3.3.1 Fully Distributed Disambiguation

A big leap forward in the development of parallel-distributed language-processing systems was made in [McClelland et al., 1986d]. By evaluating the studies done by Cottrell, Small, Waltz and Pollack, McClelland and Kawamoto conclude that only the semantical level of the latter two uses real distributed knowledge-representation techniques. The other implementations rely usually on local distributions (one unit, one concept).

To avoid the expected shortcomings of these representation techniques, a model which follows up on the work of [Anderson, 1983] and [Hinton, 1981] is proposed. Two sets of highly interconnected units are presented: one for representing the surface structure, and

one for the case structure. The network represents simple sentences (one verb and one to three NPs) as patterns of activation. After several learning iterations, the model did a good job on correction, disambiguation and generalization. Though the model lacked a connectionist parser (the input consisted of sentences in canonical form) and a lot of prewiring had to be done, it resulted in one of the first models capable of generalization and perception of language.

3.3.2 Addition of Self-Organization

One of the disadvantages of the model is the need to predefine all the semantic microfeatures, used in the distributed representation scheme, by hand. [Miikulainen et al., 1988a, 1988b] substantiate a serious attempt to avoid this problem. By connecting a lexicon with the input and output units of a back-propagating neural network, semantic microfeatures are formed within the lexicon. The automatic discovery of semantic microfeatures is even compared to the formation of topological maps in the Kohonen model.

Actually, this work of Miikulainen & Dyer can be compared to work done in [Ritter et al., 1989b, 1990]. By forcing the system to determine a representation within the lexicon, dependent on the input and output neuron, language is presented to the system in its natural context, making the formation of a topological structure possible. If a word is shown to a network, the place of the word on the map is completely determined by the coding scheme. By presenting a word in combination with its context, this problem is eliminated (see Section 3.3.7 on page 82).

3.3.3 Derivation of Semantics for Case Role Assignment

Another effort to derive semantics for case role assignment and sentence comprehension can be found in [St. John et al., 1987, 1988a, 1988b].

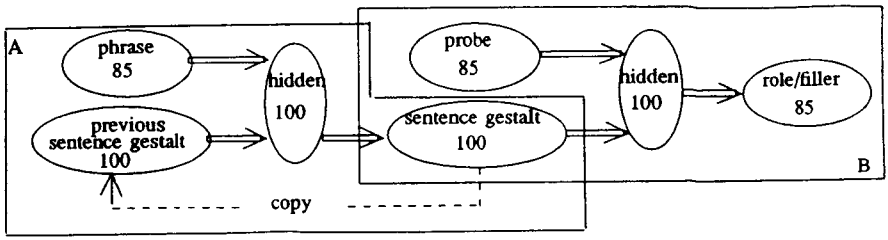


FIGURE 6. The architecture of the network (reprinted from [St. John et al., 1988]).

Here, phrases feed into a recurrent (output feed-back, like [Jordan, 1986]) localist back-propagating network with some hidden units, result in output-patterns forming a sentence-gestalt. By using this output patterns in combination with the probe as input for the next network, sentences are represented in their proper context, making semantical derivations like slot/filler disambiguation and case role assignment possible.

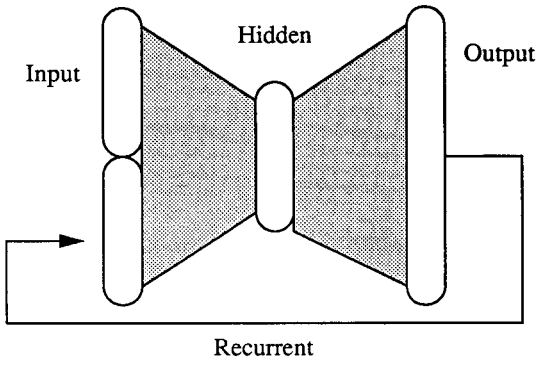


FIGURE 7. Jordan's recurrent network: feed-back of the output units into the input units.

3.3.4 Dynamic Binding & Distributed Data Representation

Although distributed representations seem biologically more plausible than local representation schemes, the first method has shown less impressive results in NLP than the latter one. Lange and Dyer state that the inability of distributed neural systems to handle natural language input of complexity higher than extremely simple sentences is mainly caused by the lack of a dynamic role-binding mechanism that propagates these binding constraints during inferencing. By adding the necessary mechanism to a localist spreading-activation model, plan/goal analyses of the input are represented in a highly-activated path in the network, thus resulting in a model better capable in inferencing, planning, schema instantiation, word-sense disambiguation and dynamic re-interpretation tasks required for NLP [Lange et al., 1989].

3.3.5 PARSNIP

Hanson and Kegl developed a parsing system that is based on the auto associator version of back-propagation. This is, in this network the input values and the output values are the same. By doing so, the hidden layer forms a fully distributed internal representation of the data set. The model used 270 input, 45 hidden, and 270 output units. By using fewer hid-

den units than output units, the model is forced to derive a more compact (distributed) structure than the surface structure. Sentences are presented in parallel, resulting in a maximum length (15 words) parser.

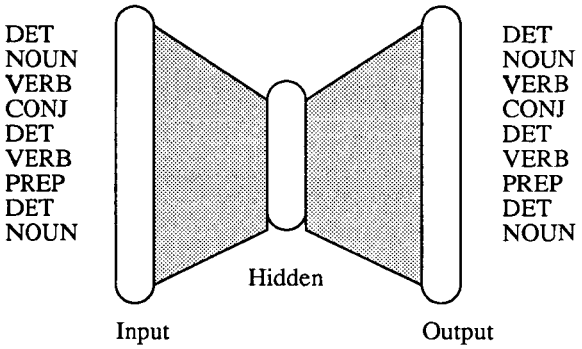


FIGURE 8. The PARSNIP auto associator network.

The authors tested the model on the 1 million sentence Brown corpus [Francis et al., 1979]. Only the syntactic categories were presented to the model, not the words itself. Up to 1000 sentences from the corpus were presented to the model. In the case of 10 and 100 sentences, the model achieved a performance rate of 95%. When confronted with 1000 different sentences, PARSNIP obviously reached its capacity and performed no better than 85%. Generalization was tested by presenting respectively 10, 100 and 1000 novel sentences to these three networks. In the first case 50% recognition was achieved, in the second 60% and in the final case 85%.

The model could handle incomplete sentences, garden path sentences and single-center embedded sentences. Hanson and Kegl did not see PARSNIP as an adequate model for language acquisition because it lacked a notion of structure. However, it is very interesting as a simple corpus-based natural language processing model [Hanson et al., 1987].

3.3.6 The Simple Recurrent Network (SRN)

In [Elman, 1988], the author uses a recurrent network, feeding back the hidden layer units into the input layer (this in opposition to Jordan, who feeds back the output layer). By giving the first and second element in a string, the network can predict the third one. Elman uses this capability to learn the network simple sentences. After learning it was capable to

determine the grammatical correctness of a sentence. The system could even categorize words into syntactical groups, without any notion of grammar.

The automatic determination of linguistic results obtained by Elman has been analyzed by many other researchers. [Servan-Schreiber et al., 1988, 1989, 1991] and [Cleeremans et al., 1989, 1990] showed that these networks can determine the grammatical correctness of finite state grammars. Finite state grammars are among the simplest grammars: the appearance of a symbol in a string is determined by the precedents in that string. However, natural language is known to be right-side (strings to come in the near future) dependent too.

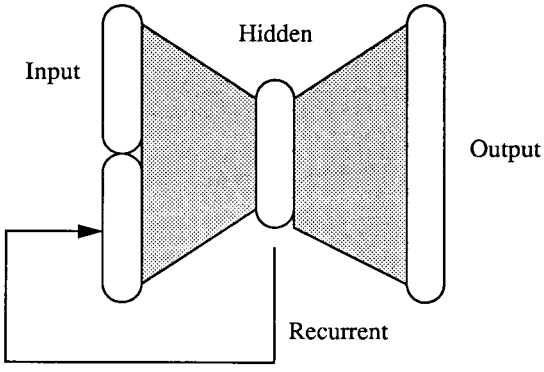


FIGURE 9. The Simple Recurrent Network (SRN): feed-back of the hidden units into the input units.

In [Allen, 1990] and [Stolcke, 1990] the authors demonstrate even more properties and functional capabilities of networks with comparable architectures, learning rules and computational power. Although these networks embody the usability of recurrent neural networks for serial processing, finite-state grammars are insufficient for realistic natural language processing.

It is proven that Elman's model is more powerful than Jordan's model in [Cottrell et al., 1989]. Here the authors show that there are certain tasks such as counting that cannot be taught to a Jordan-style network whereas a Elman-style network can learn this behavior up to a certain limit.

However, the SRN reveals unstable behavior if the training set or the neural network becomes too large. Elman introduced a method which he called *incremental learning*

[Elman, 1991b]. Here he suggests one should start with a small training set that slowly grows toward a more realistic representation of the real problem.

Somehow, Elman has not completely solved the problems occurring in his model. First, there is much critique on the SRN because it ignores structure completely. Next, large SRNs still have instable characteristics despite attempts from Elman to prove otherwise. Finally, the SRN cannot handle recursive structures, resulting in an explosive growth of the number of needed neurons when one tries to implement center embedding [Elman, 1991a]. This growth is mainly due to the fact that the SRN represents the recursive behavior as a finite state machine (FSM). To do so, it needs an exponentially growing amount of neurons with respect to the number of states in the system.

3.3.7 The Semantotopic Map

In [Ritter et al, 1989b] and [Ritter et al., 1990] sentences are presented to the system as a vector concatenation of words (X_s) with their corresponding contextual structure (X_c). Representing single words without context has no meaning in the Kohonen model. The assigned codes are arbitrary, and therefore of crucial influence on the derivation of organization in the map: by assigning another code to an object it is placed on a different position of the map. However, if a semantic map can be derived, by showing the words in their proper context, then the individual encodings become irrelevant.

Although the Kohonen model offers interesting results in language acquisition, it does not provide a complete model for language acquisition because of the inability of the model to derive and to process language structure.

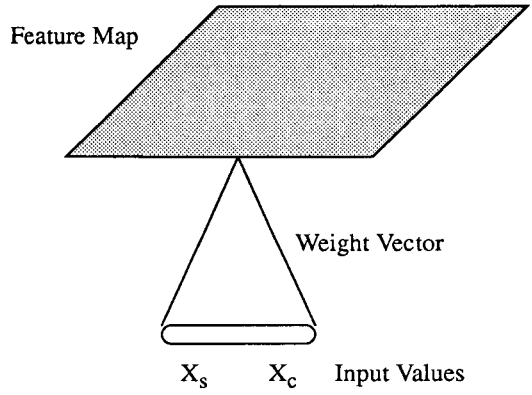


FIGURE 10. Semantotopic feature map.

[Jagota, 1990] shows another associative memory application of neural networks: a lexicon is implemented in a Hopfield Network. The known advantages of neural networks, such as incomplete retrieval, error correction and generalization were all observed.

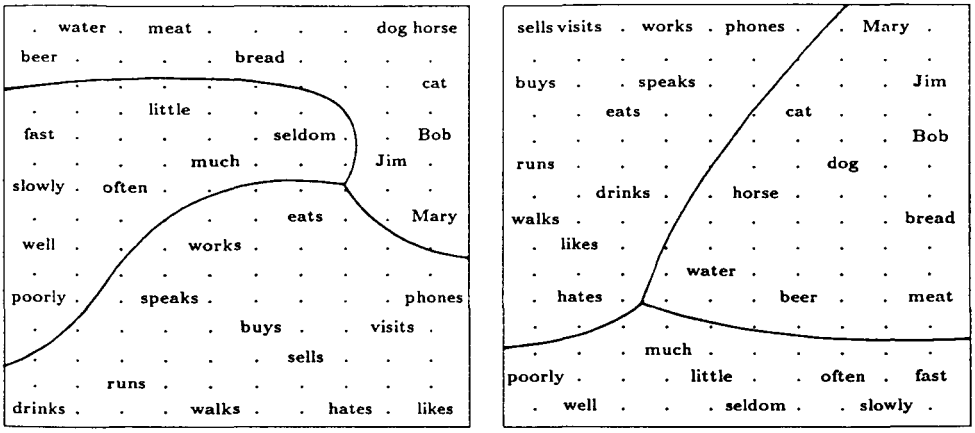


FIGURE 11. The semantotopic map for input manually tagged with contextual information (left) and the map for a restricted context of the immediate predecessor only (right) (reprinted from [Ritter et al., 1989b]).

Related to the above described recurrent neural networks, is the problem of language generation. [Kamimura, 1990a-b] shows that a recurrent neural network can generate arbitrarily long sentences, where [Williams et al., 1989a-b] have show instabilities. A variable learning rate is added to the back-propagation algorithm, and the Minkowski-r power metrics are used instead of an ordinary error function. The sentences remain however simple (finite state grammars). [Kukich, 1988] shows the relevance of connectionism in language generation. Input is fed into a time-delay network (also called running text in an input window). By back-propagating input/output pairs linguistic transformations, sememe-to-morpheme and morpheme-to-phrase transformations are learned correctly by the network.

3.3.8 BoltzCONS

A model that combines distributed data representation *and* dynamic symbol binding is BoltzCONS[1]. Although it uses a distributed representation, it can implement structures by using a functional analog of linked lists. Due to the distributed representation, associative retrieval can be implemented easily. Objects are represented as collections of superim-

1. The name is derived from a combination of the *Boltzmann* machine [Hinton, 1981] and the name of the first MIT LISP machine: CONS.

posed activity patterns enabling the model to create new structures dynamically [Touretzky, 1990].

Touretzky's main objective of the BoltzCONS model was not to implement an alternative LISP machine, but to show that neural networks can indeed implement dynamic binding, despite the argumentation of Fodor and Pylyshyn [Fodor et al., 1988].

The objective of BoltzCONS is to implement these two properties without the loss of general characteristics of neural networks such as distributed data representation and parallel processing.

The BoltzCONS model consists of an associative memory (implemented by a Boltzmann machine) and an external control mechanism. As this may sound like an weak aspect of the model, Touretzky claims he could easily build a neural finite state machine implementing the same functionality as the external controller. However, the internal representations used to implement the linked lists are much more interesting.

In Touretzky's system, each CONS operation creates a CONS cell which holds two pointers. One to the CAR and on to the CDR of the list. In LISP the CAR of a list holds the first element, the CDR the entire list except for the first element. By applying these operations properly one can construct recursive structures dynamically.

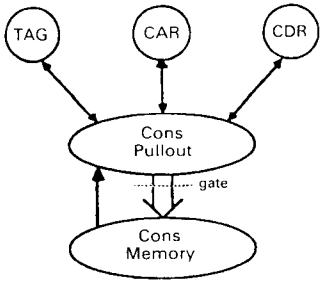


FIGURE 12. The BoltzCONS model (reprinted from [Touretzky, 1990]).

The BoltzCONS model contains five modules (see figure 12, "The BoltzCONS model (reprinted from [Touretzky, 1990])."). The CONS cells are stored in the CONS (associative) memory. The entire system supports 25 different symbols. Every CONS cell has three symbols attached to it, a TAG, a CAR and a CDR, each storing one of the 25 symbols. Each of these symbols is represented by a distributed code. The CONS pullout (the external control mechanism from the previous paragraph) is used as a gateway to the CONS triples in the memory.

As an example, Touretzky gives the following representation for a stack:

(p A g) (g B r) (r C s) (s D nil)

For every CONS element (or tupple), the last element is the same as the first of the next tupple. The element (p A g) can be removed from the stack by loading *p*, *A*, and *g* into the tag, CAR and CDR fields, so (p A g) is represented in the CONS pullout space. Next, by inhibiting the connections between the units of the CONS pullout space and the memory elements activated, an element is removed from the stack. Tupples can be moved to the stack by presenting an element to the CONS pullout and running stimulated annealing to find the energy minimum state in the memory for this object. The actual push and pop mechanism is implemented by the external control mechanism, which copies the tag space of one tupple to the CDR space of another.

Touretzky even shows how to implement trees with his BoltzCONS, by using the CAR and CRD fields as pointer to left and right tree branches. However, the model works quit artificial and the question whether it is really simple to implement the external control mechanism in a neural network is not as clear as Touretzky states. Nevertheless, BoltzCONS is the first neural model capable of solving constraint satisfaction problems with a distributed massive-parallel memory model.

3.3.9 Tensor Products

So far, most fully distributed connectionist NLP models have completely ignored the structural aspects of language. PARSNIP implements a binding between words and their syntactic category, the SRN does not implement any notion of structure (other than an instable finite-state structure in the hidden layers), and Ritter uses nothing more than a shifting window and very basic syntactical structures to derive language categories. However, it is common knowledge that *language has structure.*

Smolensky has always been an advocate of structure in connectionist models [Smolensky, 1988]. The models proposed by him and Dolan implement value/variable binding through a tensor product representation.

Tensor products are used in *multilinear algebras* [Gellert et al., 1975]:

> "Multilinear algebra is study towards multilinear forms, which are generalizations of linear forms. A multilinear form on a vector space *V* is a function that associates with any *r* vectors a number and is linear with respect to each variable. This means that if any *r-1* vectors are fixed, the mapping so defined is linear in the last vector. A *tensor space* is the generalization of a vector in linear algebra."

If one has two vectors: **v** and **w** respectively n and m dimensional, than the tensor product $\mathbf{v} \otimes \mathbf{w}$ between them is simply the nm dimensional vector with as elements all possible products $v_i w_j$ of an element from **v** and an element from **w**. This notion can be generalized easily toward higher rank tensor products.

If the string S of length n has $\{r_1,r_2,r_3,r_4,...,r_n\}$ as roles for each element $\{f_1,f_2,f_3,f_4,...,f_n\}$ in the string, then each filler/role binding can be represented by a tensor product. Instead of training a network with vector concatenations of role/filler combinations, one trains them with tensor products.

This model is mainly theoretically evaluated because it is difficult to apply in the real world. Large applications are not known yet because:

- The tensor products become very large for deep hierarchical structures, resulting in very high dimensional vectors, which are difficult to teach to a neural network.
- The mapping from vectors to structures and the other way around is a complicated process.

However, this model is the first structure-to-vector mapping that can be reversed. In addition, the technique is complete and transparent in the mathematical sense of the words, which means that a structure results in a unique vector and the other way around. This cannot be said about many other models used in connectionist NLP [Dolan et al., 1988, 1989], [Smolensky,1990].

3.3.10 Recursive Auto-Associative Memories (RAAM's)

Just like the tensor product solution, Pollack derived a method to implement structure in a connectionist NLP system. In addition, Pollack needed some mechanism that automatically derived the internal coding of structures, as he did not want to prewire everything as in the case of microfeatures.

Two separate mechanisms between symbolic structures and numeric vectors can be distinguished: a *compressor* and a *reconstructor*. The compressor encodes two patterns of equal size into one numerical vector. All terminals can be compressed in the same manner to single codes. This process can be repeated recursively by compressing the encoding of the left and right hand side branches.

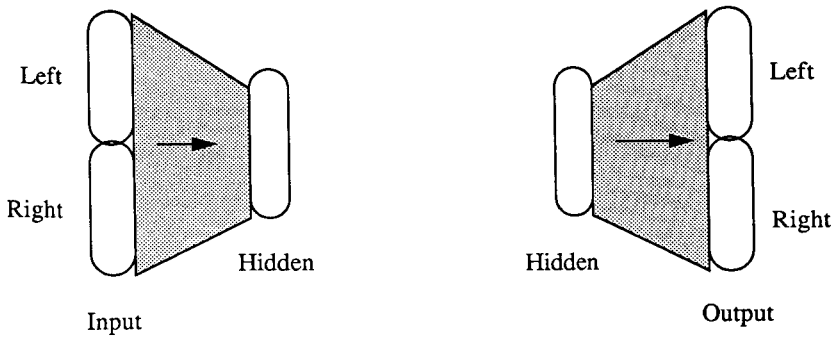


FIGURE 13. RAAM compressor (left) and reconstructor (right).

The compressor and the reconstructor can be implemented in a single feed-forward back-propagation network. The hidden layer is used to derive the compressed encoding. The outputs of the hidden units can then be used to compress encodings of higher tree branches.

Imagine the binary tree (see figure 14, "Binary tree to be encoded in a RAAM."). For this tree ((A B)(C D)), first (A B) is encoded in R_1. Next (C D) can be encoded in R_2. Finally, the representations R_1 and R_2 formed in the hidden layer can be compressed into R_3 (see Table 1, "RAAM training process").

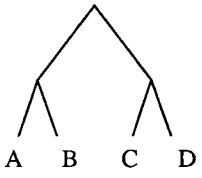


FIGURE 14. Binary tree to be encoded in a RAAM.

TABLE 1. RAAM training process

input	hidden	output
(A B)	$R_1(t)$	(A'(t) B'(t))
(C D)	$R_2(t)$	(C'(t) D'(t))
($R_1(t)$ $R_2(t)$)	$R_3(t)$	($R_1(t)'R_2(t)'$)

Finally, all tree-branch representations are stored in the same back-propagation network. This model can also be generalized to non-binary trees.

Currently, this model is under investigation at many research sites. One of the early results is that RAAM tends to work for few, simple structures, but that the model gets complicated when used to represent many different structures [Pollack, 1990].

4.0 Neural Networks in Information Retrieval

Just as in the evolution of connectionist natural language processing systems, a paradigm shift from local representations to the more distributed representations can be observed in the connectionist information retrieval models, although this general trend is a little behind on the NLP models.

In the remaining part of this section, the best known connectionist IR models shall be discussed with respect to their data representation properties. A good overview of early connectionist models in IR can be found in [Doszkocs et al., 1990].

4.1 Information Retrieval

It can be stated that Information Retrieval (IR) is the ultimate combination between Natural Language Processing (NLP) and Artificial Intelligence (AI). On the one hand there is an enormous amount of NLP data that needs to be processed and understood to return the proper information to the user. On the other hand, one needs to understand what the user intends with his or her query given the context of the other queries and some kind of user model.

However, the study of Information Retrieval has always been much more related to statistical pattern recognition than to symbolic AI techniques. The main reason is the quantity of information to be processed. Symbolic techniques would simply be much too slow. Therefore, the quality of query processing never really rose above the level of simple pattern matching without any understanding or perception of the text or the user at all.

Information retrieval can also be a very frustrating area of research. Whenever one invents a new model, it is difficult to show that it works better (qualitative and quantitative) than any previous model. The addition of new dependencies often results in a much too slow system. Systems such as Salton's SMART exist for over 30 years without having any serious competition [Salton et al., 1968, 1973, 1983a-b,1985, 1987, 1988a-b, 1991,1989], [Salton, 1968, 1971, 1972, 1980a-b, 1981, 1986, 1989].

4.1.1 Techniques used in Current Information Retrieval Systems

There are a number of methods in information retrieval that are difficult to beat.

- Adjacent character statistics or *n-gram* analyses. A window of size *n* is shifted over a text. For every possible character combination in a stored document, the frequency is kept. By comparing the frequency vector of a query to that of all documents in the data base, a measure of correlation can be calculated [Forney, 1973] [Hull et al., 1982]

[D'Amore et al., 1988] [Kimbrell, 1988]. The main advantages of this method are highly robust behavior and the elimination of a dictionary. As a result, only a global surface analysis of the text is obtained.

- *Inverted indices* (representing the exact position of every content word in a stored document) are a well known, accurate and fast technique [Sparck Jones, 1971]. Retrieval with inverted indices can be extended to boolean queries (A *and* B) and adjacent queries (A *within 5 words of* B). The difference between single or multiple occurrences in one document is not measured; therefore, ranking the retrieved documents on basis of their relevance is not possible.
- Next, by giving weight values to the index terms, a so-called *relevance ranking* algorithm can be designed. Hereby, the relevance value of a document is calculated by multiplying the total occurrence of an index term in a document by a relevance factor. Normally, frequently occurring words have low relevance factors, seldomly occurring words have high relevance factors. The documents are ranked in order of their total relevance value. The weights can then be determined manually or automatically. In general, these weights are normalized with respect to the document size [Cooper, 1971].
- Once a certain query is processed, the user can feed the results of a query back into the system. If the user indicates how good or how bad a certain result was, the computer can change the results according to the users input. This technique is called *relevance feed-back* [Salton, 1983]; it is known to be very effective.
- Instead of using the original word in the index, one can use a subset of artificial (semantic) entities. Each natural word is categorized into one semantical group. The index term only contains the semantical notions: *latent semantic indexes* [Dumais et al., 1988]. This type of models can be extended to conceptual instead of semantical items, called *conceptual information retrieval* [Mauldin, 1991]. Both techniques can best be seen as an addition to standard inverted (weighted) indices. Due to its manual nature, the maintenance of semantical groups (or concepts) is expensive. Moreover, semantical groups are subject to personal preferences and once a word is categorized into a certain group, it can no longer be retrieved from others.
- Quite useful is the addition of a (layered) *thesaurus* to the system in order to provide the user with (semantical) alternatives for key words in his query. The addition of a thesaurus can be a very powerful solution for restricted domain applications.

All these techniques are fast, surprisingly accurate and therefore difficult to beat. Another common property is that most of them have been invented over 30 years ago.

4.1.2 Precision and Recall

Since the beginning of the information retrieval research, the measure of quality has been a severe problem. The correct result of a query is difficult to indicate exactly because it is always subject to personal preferences. However, two basic notions are in use: *precision* and *recall.*

- Precision indicates the number of correctly retrieved documents relative to the total number of retrieved documents.
- Recall indicates the number of retrieved documents relative to the total number of related documents in the data base.

In general, these two values are said to be inversely proportional, which means that an increase in one of them results in a decrease of the other. Some indicate the performance of their systems by means of these values. Others compare their system to a well known system. There also are a number of standard test databases which are completely hand-analyzed. The main problem here is that these data base are quite small (less than 1 Mega byte, which is absolutely insufficient for proper comparisons of statistical models).

Well known in this context is the study by Blair and Muron. A large number of lawyers were confronted with a legal information system. They were asked to continue searching until they *thought* 75% of all relevant cases were found. As it turned out afterwards, most of them found no more than 25% [Blair et al., 1985].

In general, these problems are caused by the fact that:

- relevance is a highly subjective concept,
- relevance is always relative to the other documents retrieved, and
- most users don't know what they are looking for.

As a matter of fact, the most successful queries are the ones in which new answers to new questions are given, a highly difficult task.

4.1.3 Current Problems in Information Retrieval

The main objectives of current IR research can be characterized as the search for systems that exhibit adaptive behavior, interactive behavior and transparency. In more concrete words, these models should implement properties for:

- Understanding incomplete queries or making incomplete matches,
- Understanding vague user intentions,
- Ability to generalize over queries as well as over query results,

- Adopting to the needs of an evolving user (model),
- Allowing dynamic relevance feed-back,
- Aid for the user to browse intelligently through the data, and
- Addition of (language) context sensitivity.

In addition to these research topic there is an increased interest in the integrating of traditional databases, free-text systems, multimedia and the addition of (language and common-world) knowledge.

Considering the general properties of neural networks, one can (intuitively) understand the common ground both disciplines share. Neural networks exhibit robust, adaptive, generalizing and context sensitive behavior in comparable pattern recognition tasks. However, the application of neural networks in information retrieval will probably not be so easy as it seems at first sight.

4.2 Localist Information Retrieval Models

As in connectionist NLP, the first connectionist IR models were based on local connectionist models. In general, new neural models are developed in the context of connectionist NLP. Afterwards, these models are often applied in IR.

4.2.1 Information Retrieval as an Interactive Activation Model

Credited for the first application of neural networks in information retrieval is Michael Mozer. He was particularly interested in the IR systems used in bibliographic search.

The model consists of two sets: a set of descriptors and a set of documents. The descriptors are in fact index terms that can be derived from the documents automatically. As the model is used, the user query activates the descriptor units which in their turn activate the

relevant documents (see figure 15, "Mozer's interactive activation model in information retrieval.").

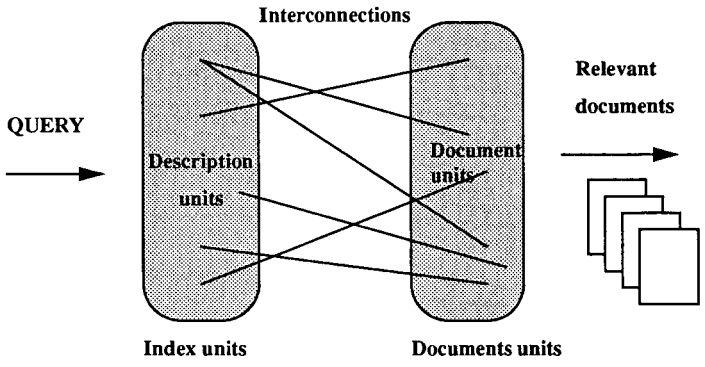


FIGURE 15. Mozer's interactive activation model in information retrieval.

The neural activation level is used to indicate the document's relevance ranking. One can doubt the relation with a neural network, however, the model shows some interesting properties caused by the spreading activation [Mozer, 1984].

4.2.2 The AIR System

The AIR (Adaptive Information Retrieval) system was the result of a Ph.D. study by Rik Belew. The model has the same structure as that of Mozer: documents and terms are represented as nodes and associations between them are represented as weighted connections. Whenever a query was set to the index terms, the related (or connected) document units would be activated.

New to Belew's model is the ability to learn (through a localized reinforcement rule). His learning algorithm has three main properties:

- The addition of new documents in the data base results in the automatic adjustment of the connections between index terms and document terms because the total sum of the outgoing weights is kept constant. As a result, more frequent index terms yield a lower activation level.
- The model is well aware of *contextual relations* due to the adaptive character to document structures. Words frequently occurring in each others neighborhood would increase each others activation level.
- Users can indicate whether they like or dislike the retrieved documents. As a result, the weights between the nodes is adapted. This highly interactive *browsing* method is a very efficient relevance feed-back method.

Belew's work is considered as one of the most comprehensive in early connectionist information retrieval, in particular because he succeeded in positioning neural networks in IR in a very successful way [Belew, 1986, 1987, 1989][Belew et al., 1988]

Other comparable early models of connectionist IR models can be found in [Personnaz et al., 1986], [Cohen et al., 1987], [Bein et al., 1988], CRUCS [Brachman et al., 1988], ZZENN [Cochet et al., 1988].

4.3 Sub-Symbolic Information Retrieval Models

As mentioned in the introduction, information retrieval systems generally use no more than global surface properties. Therefore, the need for a more content-sensitive methodology exists from the early beginning. A normal first step is the addition of semantic information to document collections. A next logical step is the combination of an inverted index with a semantical network, resulting in a hybrid model having the common integration problems known from symbolic AI.

By designing an additional semantic network, the localist neural model implements some kind of hypertext system.

4.3.1 SCALIR, a Hybrid Symbolic & Connectionist Models in Legal Information Retrieval

Belew's AIR did not use any real semantical (sub symbolic) information. As a result, retrieval results lack any deep structure. In SCALIR (Symbolic and Connectionist Approach to Legal Information Retrieval), Rik Belew and Daniel Rose extend the AIR model with a semantical network (see figure 16).

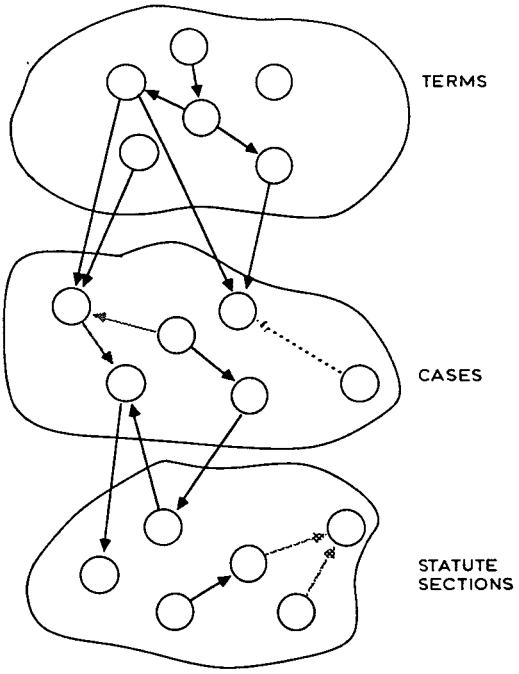


FIGURE 16. The SCALIR architecture. Nodes are represented by circles, solid lines are connectionist links, dotted or shaded arrows symbolic links (reprinted from [Rose et al., 1991]).

A clever activation function combines the proper connectionist and symbolic links. The connectionist links are called C-Links (these are the same links as used in the AIR system), the symbolic links are called S-Links. Units can be connected by either C-Links or S-Links. Both links participate in the activation value of the unit (see figure 17, "The S-links and the C-links used in SCALIR (reprinted from [Rose et al., 1991]).").

Training of the network connectionist is done with a similar rule as in the AIR model, however, instead of keeping the total of all outgoing connections constant, a more delta-rule is used to adopt the weights.

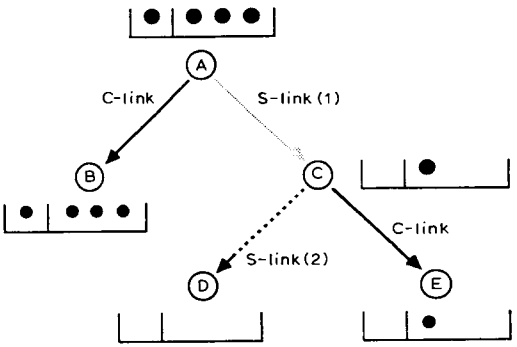


FIGURE 17. The S-links and the C-links used in SCALIR (reprinted from [Rose et al., 1991]).

As with the individual work of Rose, the model does not add much to the neural sciences. However, it is very successful in mapping the notions used in the information retrieval to those of the neural network community [Rose, 1990, 1991], [Rose et al., 1989, 1991]

4.4 Fully Distributed Information Retrieval Models

As local and subsymbolic models are known for their implementation and maintenance problems, a shift towards the fully-distributed connectionist IR models can be seen in the early 1990s. However, IR is known to be a field of *very large* data bases. Therefore, the common connectionist ideas appeared to be difficult to implement as one can read in the following sections.

4.4.1 Categorizing Documents with a Back-propagation Network

Often, information retrieval has been presented as the mapping from queries to relevant documents. A simplified case of this mapping is the mapping of documents in predefined categories. Whenever documents are categorized into categories, retrieval is expected to be easier and faster. In general, this categorization is quit time consuming and therefore an expensive, manual process. Because the back-propagation network is known to be a real achiever in such non-linear mapping problems, it has been applied several times.

Documents can be represented by inverted indexes, semantical concept vectors and other representation schemes. The n-gram representation is known to be robust and easy to use. David Mitzman and Rita Giovannini used the n-gram document representation to train a back-propagation neural network the mapping between documents and categories.

Over 5,000 document vectors were manually classified into all possible categories. Every document was represented by the frequencies of the bi-gram vector (all occurring two letter combinations). The training set consisted of structured [bi-gram vector, category] pairs. After training the model was capable to classify the test set of 25,000 documents 95% of the cases into the proper categories [Mitzman et al., 1990].

The main problem in this approach is the difficult scalability of the bi-gram vectors to e.g. trigrams (the addressing space grows exponentially, resulting in neural networks of unacceptable sizes). In addition, although n-grams are known to be fast and robust, they provide no more than a simple surface analysis. Due to the supervised nature of the model, the automatic derivation of new categories is not possible (which was no problem in the application the models was used for).

4.4.2 Clustering Documents with a Simple Recurrent Network

As in the previous model, Stefan Wermter classifies documents into categories. However, as Mitzman and Giovannini use only known categories, Wermter prefers an unsupervised model that is capable to derive new categories on the spot.

A simple recurrent network (SRN) is used. In order to avoid the combinatorial explosion of the needed number of neurons, only the words in the titles are used to categorize the documents.

In a way, this work is closely related to the work done by Elman himself on the unsupervised categorization of linguistic objects. Instead of simple sentences, Wermter used titles and a clever encoding scheme [Wermter, 1991].

Here too, the restricted word set and the use of titles only limit the model's capabilities. Due to the explosive growth of the neural network, scalability appears to be quite impossible.

4.4.3 Clustering Documents with a Self-Organizing Feature Map

Another clustering effort can be found in the work of Xia Lin. Kohonen feature maps are known to cluster related objects (given some features) into related categories. In information retrieval, documents can be clustered in related groups, so retrieval of associated documents can be facilitated.

In [Lin et al., 1991], the authors train a Kohonen feature map with a number of vectors derived from a set of scientific documents. A predefined set of words is selected. The relative frequency of each of these words is represented by one dimension in the feature vector. Next, depending on the words used in the document titles, a feature vector representing the document can be obtained. These vectors are used in the training phase.

Because titles holding the same words are probably discussing the same subject, they should be represented by neurons in neighboring regions, yielding a common subject (see figure 18, "A self-organizing semantic map (reprinted from [Lin et al., 1991]).").

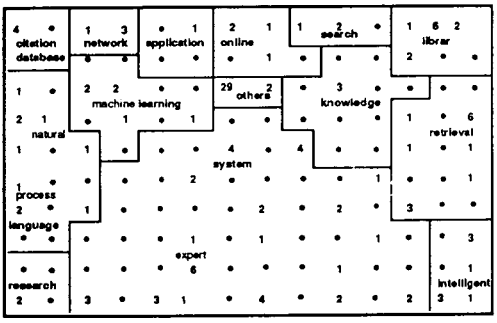


FIGURE 18. A self-organizing semantic map (reprinted from [Lin et al., 1991]).

This work too has a number of unsolved questions:

- Clustering is known to be expensive, in particular in dynamical environments. This holds also for Kohonen feature maps, which demand longer training times than straightforward statistical techniques.
- Kohonen feature maps can cluster small amount of data. In the case of large data collections, the feature maps can entangle easily.
- The data is clustered on a preselected number of words occurring in the title only. The model probably collapses if one uses abstract-only or full-text clustering. Moreover, as almost anywhere else in IR, only global surface properties of the titles is used, no structure or meaning whatsoever is involved.

4.4.4 Learning Query-Documents Set to a Back-propagation Network

As mentioned before, information retrieval can be seen as the mapping from queries to relevant documents. K.L. Kwok collected such large sets of query-documents pairs from the use of a large IR-system. The query as well as the documents were represented by vectors in which every dimension represented the weighted occurrence of an index term.

Next, he trained these vector combinations to a back-propagation neural network. As a result, the model is capable to generalize unknown queries to known document (sets) [Kwok, 1989, 1990, 1991].

This model really added some value to connectionist IR systems. First, Kwok uses his neural network as a generalizer for unknown queries. He is not interested in clustering or category derivation, but only in document retrieval. By doing so, he implements an interesting mapping in a back-propagation network. In fact, if one should use this network in addition to known techniques such as inverted indexes, relevance ranking and relevance feed-back, very interesting new properties can be observed. However, the number of possible query-document pairs grows exponentially with the size of the data base. Therefore, it is difficult to scale to larger systems.

4.4.5 Knowledge Representation in Information Retrieval with a Simple Recurrent Network

All models discussed so far used global surface analyses only. The relation between semantical networks, information retrieval and neural networks could well be observed in the local connectionist IR models. Due to the lack of structural representation schemes, semantic networks were not applied in fully-distributed neural IR models until Bob Allen used a simple recurrent network (SRN) for such an application.

The addition of semantics is considered essential for the retrieval results. However, there are two main problems. First, they have to be added by hand. Secondly, this often slows down the system to an unacceptable rate.

Allen trained a SRN with pairs of question and answers. The questions were composed of multiple entities such as "whois motherof mary". Answers were single objects like "sue". A possible training sequence would then be: "whois motherof mary * sue", where * indicates the separation between a question and an answer. Over 32 input terms, 27 output terms and, 8 persons were composed to a corpus of 626 sentences. After training the model was capable to answer various simple questions [Allen, 1991].

Allen hoped to apply his findings in a information retrieval system as a module to add highly structured semantical information to flat textual surface properties. The scalability of this method may be doubted for large data bases. Partly due to the complex relations obtained as the data base grows and partly due to the explosive growth of neurons in the SRN as the questions become more complex and larger in number.

5.0 Conclusions on Connectionist NLP and Connectionist IR

Connectionist NLP and connectionist IR both suffer from certain limitations. For reasons of presentation only, they will be discussed separately in the next two sections, although some limitations are shared.

In general NLP suffers much more from the limitations pertaining to the representation of structural information in neural networks than IR models, due to the fact that almost none of the IR models uses structural and semantical information. In addition, IR suffers much more from the limited memory capacity of neural networks than NLP because IR data bases are typically an order of magnitude larger than (current) NLP corpora. However, as NLP corpora become larger and IR systems need to apply structural information, both kinds of systems will eventually share the same properties as described below.

5.1 State of the Art in Connectionist NLP

Language has structure. As it seems, the first connectionist NLP models were completely aware of this. Structural language aspects were implemented explicitly in the architecture of the model. As a result, the data representation used was completely local and the systems could not be trained automatically due to the complex architecture.

Without distributed data representation, the only interesting aspects of such models are the spreading activation and disambiguation properties. Error-correction and generalization cannot be observed in those models.

Therefore, subsymbolic systems appeared. Here, the structural aspects of language become less important so the model could use a more distributed data representation. Most of these systems could still not be trained automatically, unless they had extremely simple architectures.

In later, fully distributed models (like PARSNIP and the SRN) the structural aspects of language disappeared totally. The reason for this can be given easily: the implementation of *hierarchical structure*, *learning* and *distributed representations* could not be realized in one transparent system. Most research focussed on the distributed learning models.

Lately, connectionist NLP research tends to rediscover structure as an essential aspect of language. Nowadays, one can observe an increase in efforts to combine distributed data representation, learning, and (recursive) hierarchical structures (BoltzCONS, Tensor Products, and RAAMs). The *structure-to-vector* mapping problem has not been solved completely yet. However, the results obtained in recent connectionist NLP research are stimulating.

In addition one can take notice of another trend in connectionist NLP. The very first models process language in the same manner as the symbolic AI systems do: the neural network is much too often used as a processor rather than as a memory device.But, there should not be any difference between the processor and the memory, they should be the same! This more *associative* approach of NLP uses neural networks in the way they are meant to be and should produce better results than much of the models presented in this chapter.

5.2 State of the Art in Neural Information Retrieval

Two different types of information retrieval can be observed. On the one hand there are relatively static databases which are investigated with a dynamic query (*free text search*, also known as *document retrieval systems*). Next there are the more dynamic databases which need to be filtered with respect to a relatively static query (the *filtering problem* also known as *current awareness systems* [DARPA, 1991]). In the first case the data can be preprocessed due to their static character. In the second case, the amounts of data are so large that there is no time whatsoever for a preprocessing phrase. A direct context sensitive hit-and-go must be made.

As in connectionist NLP, the localist models adopt well to the models currently in use in information retrieval. Index terms can be replaced by processing units, hyperlinks by connections between units, and network training resembles the index normalization process. However, these models do not adapt well to the general notion of neural networks for the same reasons as it is the case with local connectionist NLP models.

In addition, it is difficult to imagine what to teach to a neural information retrieval system if it is used as a supervised training algorithm. The address space will almost always be too limited due to the large amounts of data to be processed. A combination of structured (query, retrieved document numbers) pairs does not seem plausible either, considering the restricted amount of memory of (current) neural network technology. Nevertheless, most of the neural IR models found in the literature are based on these principles.

Also problematic are the so-called clustering networks. Due to the large amounts of data in free text databases, clustering is too expensive and is therefore considered irrelevant in changing information retrieval environments [Willett, 1984, 1988].

More interesting are the more unsupervised, associative memory type of models, which can be used to implement a specific pattern matching task. This type of neural networks can be particularly useful in a filtering application. Here, the memory demands of the neural network only needs to fulfill the query (or interest) size, and not the size of the entire data base. It is here where neural networks are expected to be useful and relevant for information retrieval.

Chapter 3

Recurrent Kohonen Feature Maps in Natural Language Processing

"If you think to build a tower, first reckon up the cost"

-- St. Jerome

Abstract

In the 1980s, back-propagation (BP) started the connectionist bandwagon in Natural Language Processing (NLP). Although initial results were good, some critical notes must be made about the blind application of BP. Most such systems require that contextual and semantical features are added manually by structuring the input set. Moreover, these models form a poor approximation of the brain structures known from neural sciences. They do not adapt smoothly to a changing environment and can only train input/output pairs. Although these disadvantages of the back-propagation algorithm are commonly known and accepted, other more plausible training algorithms, such as unsupervised training techniques, are still rare in the field of NLP. The main reason is the high complexity of unsupervised training methods when applied in the already complex field of NLP. However, recent efforts implementing unsupervised language learning have been made, with interesting results.

Taking off from this earlier work, this chapter presents a recurrent self-organizing model (based on an extension of the Kohonen feature map), which is capable of deriving contextual (and some semantical) information from scratch. The model implements a first step towards an overall unsupervised language learning system. Simple linguistic tasks such as single word clustering (representation on the map), syntactical group formation, derivation of contextual structures, string prediction, grammatical correctness checking, word sense disambiguation and structure assigning are carried out in a number of experiments.

Although premature, the first results are promising and show possibilities for other even more biologically-inspired language processing techniques such as real Hebbian models.

1.0 Background

The importance of connectionism in Natural Language Processing (NLP) has been advocated by many researchers lately. The ability to represent knowledge in a distributed way without the addition of explicit knowledge structures and loss of generality convinced many in the field of the usability of such techniques in NLP [Graubard, 1988]. However, most neurally inspired models implement only a fraction of the knowledge acquired in the neural sciences. The main reason for this is the tremendous complexity of biological neural networks. Besides the large amount of neurons and connections needed, there is the mathematical complexity of the feed-forward and training rules.

The popular back-propagation algorithm is a very restricted example of such neurally inspired networks. There are no connections within a layer, only between layers. So, only connections between these (usually three) layers adapt. Furthermore, only a small number of architectures feature recurrent fibres. If one puts biological neural networks next to these models, just a small shadow of resemblance remains. What is known of neural structures in the cortex tells us at least that there are connections within a layer, which are learned just as the inter-layer connections are. Brain structures consist of multiple layers. Next, there are many recurrent fibres connecting neurons at different layers. All connections learn due to synaptical plasticity, there is constant adaption to a changing environment by reformations on the cortex map. The model converges to something other than a zero-point or zero-line in a multidimensional space: more realistic models do not converge in the way proposed in various models, but show chaotic behavior.

Back-propagation is a so-called supervised training algorithm where an external teacher adjusts the weights [Rumelhart et al., 1986a]. Unsupervised training rules (without an external teacher) have been developed in different directions. All of them are based on a competitive training principle: if a pattern fits best on a certain region on the map, adjust the weights so it fits even better on this position the next time it is presented [Rumelhart et al., 1985]. These rules, in turn, are all derived from the Hebbian training rule: if two neurons are activated at the same time, increase the connection strength between them.

Overall one observes three main streams of research efforts [1]. First, there are the competitive algorithms of Grossberg and Kohonen. Grossberg uses a two-layer, fully interconnected model based on competitive training principles: Adaptive Resonance Theory ART [Grossberg, 1980, 1988] [Carpenter et al., 1988]. Kohonen developed a one-layer map where all input sensors are connected to all neurons. The map does not fire but results in the formation of a topological map of the input sensor values: a Self-Organizing Map.

1. The simple recurrent network (SRN) by Elman is also known to be unsupervised. However, it is based on a centrally controlled weight adaptation process. Therefore, it will not be considered unsupervised in the sense of decentralized control, which can be observed in the three main streams discussed in this paragraph.

Next, the more Hebbian models of Von der Malsberg and Linsker can be distinguished [Hebb, 1949], [Malsberg, 1973], [Linsker, 1988]. Here, the training rule adapts all connections in all directions. Finally, recent developments indicate good possibilities for evolutionary models based on evolutionary selection theories and so-called Genetic Algorithms. These models implement population theories in neural networks: successful populations multiply faster than ones that are unsuccessful (with respect to some quality measurement) [Holland, 1975] [Goldberg et al., 1988], [Goldberg, 1988]. The unsupervised training algorithms mentioned above are ordered in increasing complexity.

Unsupervised training might be defined as the total absence of a central control mechanism which implements an external teaching unit. There are different interpretations of this definition. One can abandon a central control mechanism totally or accept it at the neuronal group level. There is no evidence for central control mechanisms in the human brain. Neural sciences indicate a locally distributed organization. Specific functions are implemented in specific parts of the brain.

Moreover, locally this knowledge is distributed implementing association, generalization and adaptation mechanisms. A low level normalization process can be seen as a necessary locally specific function, implementing the overall unsupervised training process. But, it could also be abandoned completely on grounds of not being unsupervised (some central mechanism must control the normalization). The neurological plausibility of this decision has a very vague border: what is local and what is locally distributed? At first sight central control mechanisms must be avoided. All knowledge should be distributed. But if a subnet implements a specific function, different training rules and inter-layer connections can be seen as a locally specific function making that part of the network especially suitable to implement a certain function.

In certain cases the application of neurally inspired methods in NLP is obvious. In others, it can be very difficult to develop them. Spelling correction, lexical access, word sense disambiguation and generalization are implemented by relatively simple means. These problems use the most basic and implicitly present characteristic features of neural networks: association and generalization caused by the parallel distributed knowledge representation. On the other hand, solving the representation of time (or sequences needed to define grammatical correctness and to carry out a sentence parse) in parallel systems is quite a problem.

In sequential processing, contextual information of sequential strings is derived for free. However, in parallel processing one can either add explicit time marks (increasing the dimension of the input vectors), or concatenate different parts of the input towards one large vector (the window principle), or add buffers to the system, or use recurrent fibres.

While the first two options are not really expected to be found in biological systems, the second one has its roots in the seven plus/minus two window theory of human short-term memory [Miller, 1956]. But, there is a psychological problem. Windows, as implemented in current neural networks, use seven plus/minus two characters, phonemes or words as input. Results from psychology indicate that the size of a short-term memory is about seven plus/minus two, but the question is seven plus/minus two of *what*. Different people have different objects in short-term memory. Therefore a simple window mechanism on character, phoneme or word level does not suffice. The third option has not yet been worked out in relation to neural networks (at least, not without the addition of an overall control mechanism). A buffering system working at different hierarchical levels can overcome the disadvantages of window systems. However, results in this direction are unripe.

The use of recurrent fibres already had great impact in the connectionist language processing community [Elman, 1988] [Cleeremans et al., 1989] [Allen, 1990]. Several features of recurrent fibres are analyzed thoughtfully, making them suitable in an unsupervised environment [1]. Fibres as added here result in a model able to recognize languages generated by finite state grammars (FSG). However, finite state machines (FSM) alone are too restricted for complete natural language processing. The occurrence of an object in a FSG string depends only on objects encountered in the near past. One needs more powerful mechanisms, such as ones with the ability to anticipate forthcoming string elements, to implement some form of context sensitivity.

Other important questions in unsupervised (connectionist) language learning concern the need for semantical additions in the training set, the importance of negative information and the need for recursion.

From a theoretical point of view the implementation of recursion in neural networks is very interesting. However, does one really need recursion to define grammatical structures or can natural language be recalled by using association in a distributed neural network? The research carried out tries to provide an answer to these questions.

After the evaluation of some backpropagating connectionist NLP systems (Chapter 2, "Neural Networks in Natural Language Processing and Information Retrieval"), this research aims at the application of unsupervised training mechanisms to NLP problems. The current chapter describes results obtained with an extended Kohonen model. The model performs a number of linguistic tasks.

1. Kohonen feature maps and ART already have recurrent connections between the neurons within one layer. However, these recurrent connections have different characteristics than the temporal recurrent fibres as mentioned in this context. Therefore, if the term "recurrent" is used, within-layer dynamical processes are not intended.

As mentioned, the author is aware of the implicit restrictions made in the self-organizing map, but by investigating a recurrent Kohonen map, the research serves two purposes. First, it will demonstrate that the Kohonen map can be very useful in NLP and other symbolic processing jobs [Hemani et al., 1990], although so far it is mainly used in vector quantization processes and is not known for its symbolic processing abilities yet [Rubner et al., 1990]. Second, the model learns linguistic structures from unformatted strings passing by: an unsupervised training process applied to NLP.

This research is part of a study of the usability of connectionist training methods in natural language processing. This in its turn is part of a long term project developing new methods in computational-linguistic areas such as data-oriented parsing, early language acquisition, and structural/semantical disambiguation.

Parts of the results presented in this chapter have been published in [Scholtes, 1991a-1991i].

2.0 Introduction

The Kohonen formalism is a competitive training algorithm [Kohonen, 1982a, 1982b, 1982c, 1984, 1988, 1990a, 1990b]. A two-dimensional map is constructed in a rectangular or hexagonal structure from individual neurons. Each neuron has a number of input sensors with an input activation and an input weight. All neurons have the same number of input sensors. The training rule acts in the following way. First, copy the activation values of an input element into all input activation sensors of all neurons. Next, determine the best match by finding the neuron with the minimum mathematical (e.g. euclidean) distance between input and weight values. Then, adapt the weights of the neurons within a certain region of this minimum, so they will recognize the current input vector better in the near future. After numerous cycles, a topological map is formed, holding related elements in neighboring regions (see Chapter 1, section 4.11 for a detailed description of Kohonen feature maps).

Obvious applications of Kohonen feature maps in language processing can be found in [Miikkulainen et al., 1988a, 1988b], and [Schyns, 1990a-b, 1991]. Although actual training is done with a supervised training method, the Kohonen formalism plays a conceptually significant role. Another attempt to use Kohonen feature maps in NLP can be found in [Ritter et al., 1989b, 1990]. Here, words are taught to a single Kohonen map by feeding the concatenation of a symbol code and a context code into the input sensors, resulting in a so called semantotopic map [1].

To achieve automatic derivation of syntactic features as well as syntactic structures, one has to use a method similar to the one proposed by Ritter: add implicit context sensitivity to the system. There are several methods. First, the window principle, as used by Ritter, results in a restricted sentence length. In [Kohonen et al., 1981] the authors propose a centrally guided buffering mechanism to implement temporal processing abilities. Although this is a practical solution, a more distributed and unsupervised mechanism is preferred. In [Tavan et al., 1990] sensor values are exchanged between a sensor and a feature map, resulting in the formation of an associative memory. Although interesting, this is unsuitable for the current purpose. Next, [Thacker et al., 1990] describes the design of an unsupervised multi-layer context-sensitive model, which uses recurrent fibres. This is a problem in the Kohonen training algorithm: it lacks a notion of firing. [Kangas, 1990] provides a solution for this problem, making it possible to use recurrent fibres in a multi-layer Kohonen map.

Kangas calculates the degree of correspondence between input values and weight values for all neurons on the map. Every neuron is represented by a dimension of a vector. This

1. See Chapter 2: "Neural Networks in Natural Language Processing and Information Retrieval", section 3.3.7 for more details on these semantotopic maps.

vector expresses the activation of the feature map to an input vector. By averaging this vector in time, the system gets more or less sensitive to changes and noisy input. The result of this vector is fed back into the first layer as contextual information. So, the input vector of the first layer consists of the concatenation of a (randomly assigned) symbolic part and a recurrent contextual part. The output vector of the first map serves as input for the second map.

Over time, the dimension of the input vectors of the second map definitely gets too large for efficient simulations. Therefore, it is normalized and reduced in dimension. Although normalization and dimension reduction are supervised processes, they can also be interpreted as a (natural) resource use process (if one neuron uses more chemical resource to obtain a voltage increase, there is less left for other neurons, resulting in a voltage decrease) [Malsberg, 1973]. On the other hand, this process should not be necessary if enough computational power were available.

The second map is *not* needed for the convergence of the first map, but as the first map converge to a categorization of words, the second map will converge to a categorization of contextual structures. In fact, the model could also be extended with a third map, that generalizes over the contextual structures. In the experiments presented, only two layer models are evaluated.

So, by training both maps according to the Kohonen formalism, the first map forms an ordering with syntactically equivalent words in subsequent regions, and the second map holds related contextual structures in neighboring regions. All resulting from single strings just passing by.

The presented model only exhibits left-context sensitivity. However, Natural Language Processing needs something more powerful. Often, the information that has been processed in the near past provides enough context to disambiguate what follows. But sometimes one needs information from the more remote past as well as the future in order to disambiguate complex grammatical structures. The model shall not succeed in this without the addition of a buffering mechanism (or memory), capable of processing information in a way which is common knowledge in sequential processing. The main problem is the lack of understanding of a possible buffering system which is not controlled by a supervised mechanism, which would result in a bottle-neck or in an unrealistic model of human information processing. Future research has to clarify this shortcoming of the model as it is proposed here.

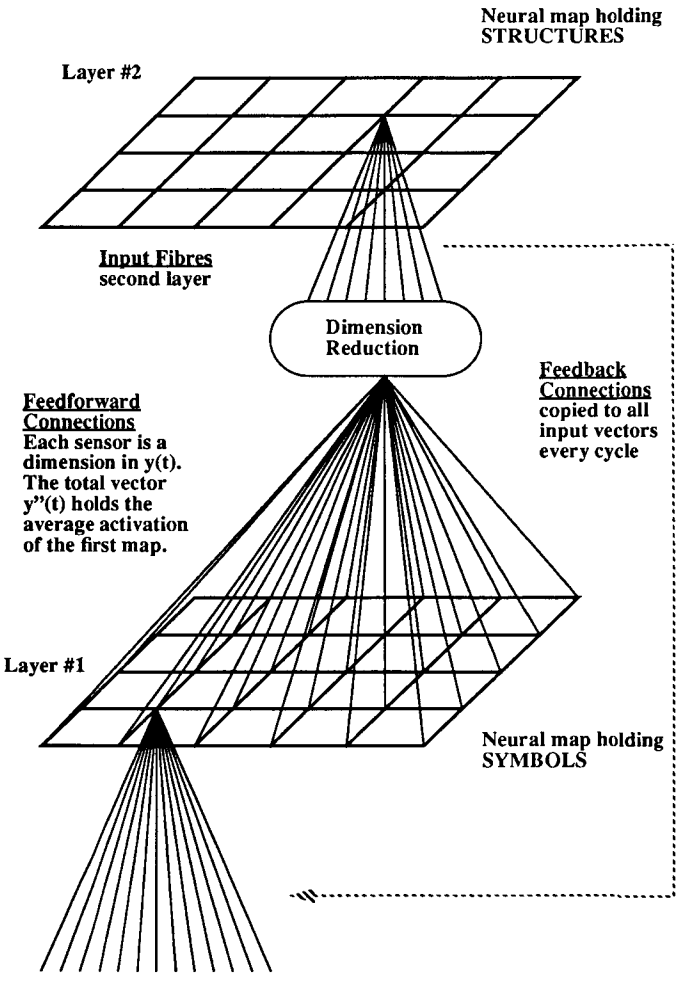


FIGURE 1. The recurrent self-organizing feature maps

In order to be useful in sentence processing systems, the neural network should at least meet the following objectives:

- Cluster (equal) words in regions of the map.
- Word class derivation on syntactical and semantical grounds.
- Predict next elements in a string.
- Determine grammatical correctness of a string (accept or reject it).
- Disambiguate word senses.
- Assign a structure to a sentence.

By combining the neural network with a conventional shell, which provides information to be processed in a convenient way, a sentence processing model is simulated where the model learns everything it knows from an unsupervised training algorithm (see figure 1, "The recurrent self-organizing feature maps").

3.0 Definitions

The following symbols shall be used to define the recurrent self-organizing model:

$N^{(1)}$	**Number of neurons in first layer**
$N^{(2)}$	Number of neurons in second layer
n_s	Dimension of symbol vector
n_r	Dimension of recurrent vector
n	Dimension of input vector
L	Layer number
$M^{(L)}$	Set of all neurons in map *L*
k or v	Number of neuron on map
$\underline{x}^{(1)}{}_s(t)$	Symbol input of the first layer
$\underline{w}^{(1)}{}_s(t)$	Symbol weights of the first layer
$\underline{x}^{(1)}{}_r(t)$	Recurrent input of the first layer
$\underline{w}^{(1)}{}_r(t)$	Recurrent weights of the first layer
$\underline{x}^{(1)}(t)$	Concatenated input of the first layer
$\underline{w}^{(1)}(t)$	Concatenated weights of the first layer
$\underline{y}^{(1)}(t)$	Activation measure of the first layer
$\underline{y}^{(1)'}(t)$	Sharpened activation measure of the first layer
$\underline{y}^{(1)''}(t)$	Averaged activation measure of the first layer
ζ_i	Vector representing dimension *i* of $N^{(1)}$
$\underline{x}^{(2)}(t)$	Input of the second layer
$\underline{w}^{(2)}(t)$	Weights of the second layer
$Dim(\underline{x}^{(1)}{}_s(t))$	n_s
$Dim(\underline{x}^{(1)}{}_r(t))$	n_r
$Dim(\underline{x}^{(1)}(t))$	$n=n_s + n_r$
$Dim(\underline{y}^{(1)}(t))$	$N^{(1)}$
$Dim(\zeta_i)$	n_r
$Dim(\underline{x}^{(2)}(t))$	n_r
$Dim(\underline{w}^{(2)}(t))$	n_r
$\underline{w} \oplus \underline{x}$	Concatenation of vectors $\underline{w}$ and $\underline{x}$

4.0 Algorithm

The training algorithm consists of 8 steps. Below, they are discussed in detail. The number between square brackets corresponds to the step of the algorithm presented at the end of this section.

First, a number of random vectors is generated to represent codes for the symbols encountered in the training set. These vectors have a dimension equal to the amount of input sensors of the first map. The input vector $\underline{x}^{(1)}(t)$ and the weight vector $\underline{w}^{(1)}(t)$ of the first layer are concatenations of the vector code for an element $\underline{x}^{(1)}{}_s(t)$ and the recurrent context $\underline{x}^{(1)}{}_r(t)$, respectively of their weights $\underline{w}^{(1)}{}_s(t)$ and $\underline{w}^{(1)}{}_r(t)$. Random vectors are substituted for $\underline{x}^{(1)}{}_s(t)$. Because the dimension of an output vector of the first map is reduced for reasons of complexity only, a second set of random vectors is generated, each representing a dimension of the recurrent input vector $\underline{x}^{(1)}{}_r(t)$. These vectors form a random basis for the recurrent input space and are indicated by the symbol ζ_i, where i is the dimension number. Later on, an exact definition of this dimension reduction shall be given [step 1].

Next, an input sentence from the training set is split into separate objects, each representing a word. An external algorithm determines unique elements occurring in the training set and assigns (at random) a code from step 1 to each of these words [step 2].

Depending on the number of training cycles, the model selects sentences from the training set at random. Its separate words are successively fed into the system. This step is repeated for the desired number of training cycles [step 3].

Steps 4 to 8 are repeated for every word in the sentence. As stated, the input vector is a concatenation of a symbolic and a recurrent vector.

$$\underline{x}^{(1)}(t) = \underline{x}^{(1)}{}_s(t) \oplus \underline{x}^{(1)}{}_r(t) \qquad \text{(EQ 1)}$$

The symbol vector is one of the random vectors generated in step 1. If the first word is fed into the system, there is no context available. Therefore the recurrent vector equals the zero-vector. Then, the input vector is the concatenation of the symbol vector and the zero-vector.

$$\underline{x}^{(1)}(t) = \underline{x}^{(1)}{}_s(t) \oplus \underline{0} \qquad \text{(EQ 2)}$$

If, on the other hand, there has been previous input, the input vector is the concatenation of the symbol and recurrent vector.

(EQ 3)

$$\underline{x}^{(1)}(t) = \underline{x}_s^{(1)}(t) \oplus \underline{x}_r^{(1)}(t)$$

Weights are always a concatenation of the symbolic and recurrent weights, otherwise previously learned information would be ignored.

$$\underline{w}^{(1)} = \underline{w}^{(1)}{}_s(t) \oplus \underline{w}^{(1)}{}_r(t) \qquad \text{(EQ 4)}$$

The result of the input concatenation is fed into the first layer of the model [step 4].

$\underline{y}^{(1)}(t)$ represents the activation measure of the first layer. If a word is the starting element of a string, the previous activation of the first layer equals the zero-vector calculating the activation value of a neuron i on the first map involves subtracting the input of neuron i from the weight of that neuron. The smaller this result, the better the neuron represents the input vector. This value is squared increased by a small value δ (to avoid dividing by zero), and inverted. So high values correspond to perfect maps.

$$\underline{y}^{(1)}(t) = \frac{1}{(\underline{w}^{(1)}(t) - \underline{x}^{(1)}(t))^2 + \delta} \qquad \text{(EQ 5)}$$

This calculation is repeated for every neuron on the first map, resulting in a vector with as many dimensions as neurons on the map. Every dimension represents the measure of correspondence between the input vector and a neuron on the map. This vector is normalized towards $\underline{y}^{(1)\prime}(t)$ by dividing its square value by the sum of the square values for all neurons on the map: $Y(t)$.

$$Y(t) = \sum_{i=1}^{N^{(1)}} (\underline{y}_i(t))^2 \qquad \text{(EQ 6)}$$

$$\underline{y}^{(1)}(0) = \underline{o} \qquad \text{(EQ 7)}$$

$$\underline{y}^{(1)\prime}(t) = \frac{(\underline{y}^{(1)}(t))^2}{Y(t)} \qquad \text{(EQ 8)}$$

As a result, the summation of all the elements of the output vector equals one. To avoid the system from being too sensitive to changes, $\underline{y}^{(1)''}(t)$, the averaged activation in time is determined by adding the activation at *t-1* and the activation at *t*, multiplying the two elements with a value ω, and *1*-ω respectively: the memory rate of the system.

$$\underline{y}^{(1)''}(t) = \frac{\omega \cdot \underline{y}^{(1)\prime}(t) + (1-\omega) \cdot \underline{y}^{(1)\prime}(t-1)}{\omega} \qquad \text{(EQ 9)}$$

A large ω results in short memory and sensitivity to changes. A small value causes the system to adapt slowly to changes in the input [step 5].

A number of vectors $\underline{\zeta}_i$ was generated randomly in step [1], one for each dimension of $\underline{y}^{(1)''}(t)$. The vector copied into the input of the second layer: $\underline{x}^{(2)}(t)$, and into the recurrent fibres of the first: $\underline{w}^{(2)}(t)$, is determined by adding the multiplications of the separate dimensions of $\underline{y}^{(1)\prime}(t)$ with the corresponding vector in $\underline{\zeta}_i$. In fact every dimension of $\underline{y}^{(1)''}(t)$ is represented by one vector $\underline{\zeta}_i$. The dimension reduction is caused by the fact that $\underline{\zeta}_i$ is of a lower dimensionality than $\underline{y}^{(1)''}(t)$.

$$\underline{x}^{(2)}(t) = \sum_{i=1}^{N^{(1)}} (\underline{y}_i^{(1)''}(t) \cdot \underline{\zeta}_i) \qquad \text{(EQ 10)}$$

$$x_r^{(1)}(t+1) = \underline{x}^{(2)}(t) \qquad \text{(EQ 11)}$$

The legitimacy of this dimension reduction is based on the heuristic that most of the elements in $\underline{y}^{(1)''}(t)$ are about zero and their norm equals 1. Therefore, this operation conserves the main characteristics of the original vector. The number of fibres is reduced enormously, caused by the reduction of the vector dimension with a factor 10. So, even large maps derive complex representations within reasonable time limits. See [Ritter et al., 1989] for a proof of the legitimacy of this operation [step 6].

Till now, the only task performed has been the feed forward of the input vector through the network. If the word in question is the first word of a sentence, it is combined with a zero-vector (representing the situation in which no context is known) and the recurrent vector equals the zero-vector. Because all string starting words have the same recurrent part, they shall form neighboring regions on the map. On the other hand, if the word is not the first

one in a string, the recurrent vector is determined by the activation measurement of the first map caused by the previous word. The map organizes sequencing words in the same region, based on equality of the recurrent vectors. (Although words on the first map keep moving until a perfect self-organization is formed, after a certain time interval positions end up being quite stable). This delicate balance between the recurrent and symbolic vector results in the desired organization (an organization in which all related elements are stored in neighboring neurons). The algorithm learns the first layer as well as the second one. The first with vector concatenations from step 1, the second with the input vector of step 6 and the weight vector of the second layer. Therefore, the model applies the Kohonen training rule. First, the best match for this element on the map is determined by calculating the minimum euclidean distance between the input vector $\underline{x}(t)$ and the weight vector $\underline{w}(t)$ for all neurons on the map.

$$v^{(L)}: \qquad \forall k \left\| \underline{w}^{(L)}{}_{v}(t) - \underline{x}^{(L)}(t) \right\| \leq \left\| \underline{w}^{(L)}{}_{k}(t) - \underline{x}^{(L)}(t) \right\| \qquad \text{(EQ 12)}$$

for k element of the feature map of layer $L = 1,2$. This minimum holds for the neuron v which represents the input vector best. This rule is applied to the first as well as to the second layer. [step 7].

If the weights of this neuron and the surrounding ones are adapted in this way, they represent the input vector better next time it occurs.

$$\underline{w}^{(L)}{}_{k}(t+1) = \underline{w}^{(L)}{}_{k}(t) + \varepsilon(t) \cdot \Phi_{kv} \cdot (\underline{x}^{(L)}(t) - \underline{w}^{(L)}{}_{k}(t)) \qquad \text{(EQ 13)}$$

$$\Phi_{rs} = e^{-\| k-v \| \cdot 2\sigma^2(t)} \qquad \text{(EQ 14)}$$

Here, σ implements the physical distance (in neurons) between the Best Matching Unit and the neurons that are updated. Neurons that are located outside this area are not updated.

Two reasons guarantee convergence towards a self- organizing state (at least, if the number of training cycles is large enough). In the first place, the training rate (the measure by which weights adapt to new values) and the region size (the number of neurons in the direct neighborhood which are adapted) decrease as a bell shaped function in time. Both variables start with a large value which decreases slowly towards zero. Therefore the changes in the weights reach zero as the number of training cycles increases towards infinity.

$$\varepsilon(t) = \varepsilon_{max} \cdot \left(\frac{\varepsilon_{min}}{\varepsilon_{max}}\right)^{\frac{t}{t_{max}}} \qquad \text{(EQ 15)}$$

Secondly, if a self-organizing state exists, it shall be reached eventually if the input is presented randomly. (Although there is no direct analogy, this effect can be compared with a characteristic of hidden Markov chains: if there exists an absorbing state and one walks randomly from state to state in the Markov model, then the absorbing state will be reached eventually. Similarly, there is no escape from self-organization).

$$\sigma(t) = \sigma_{max} \cdot \left(\frac{\sigma_{min}}{\sigma_{max}}\right)^{\frac{t}{t_{max}}} \qquad \text{(EQ 16)}$$

where

$\omega \in [0,1]$ Memory Rate,

$\varepsilon_{max} \in [0,1]$ Initial Learning Rate

$\varepsilon_{min} = [0,1]$ Final Learning Rate,

$\sigma_{max} \in [0,1]$ Initial Region Size

$\sigma_{min} \in [0,1]$ Final Region Size, and

$\|k - v\|$ Physical distanced neuron k to v

Furthermore, the larger the physical distance between a neuron and the optimum position on the map, the smaller the adjusting of the weights of this neuron [step 8].

One can expect convergence to take place on two grounds. First, the first map organizes on properties of the symbolic part of the input vector. Equal words shall be ordered in uniform classes. Secondly, there is the organization triggered by the recurrent fibres, resulting in regions of word classes which are used in the same context - not exactly syntactic classes, but something like substitutionally or semantically similar words. Both organizations influence each other, resulting in chaotic behavior of map formations. The second map follows the organization on the first map, and shall therefore only start to get organized as the first map has reached some initial state of self-organization. Convergence times shall be quite long, considering the nature of two such self-organizing processes. Furthermore, because the symbolic and recurrent vectors have the same norm, neither of them can turbo-charge the convergence process. One has to await the moment where every element is in its correct position according to the symbolic as well as the contextual constraints. Only then, convergence towards the self-organizing state occurs.

The table on the next two pages provides the reader with an overview of the algorithm.

Step	Description
1	*Generate random codes for the symbol representation:* $\underline{x}^{(1)}s$ *and the dimension reduction vectors:* ζ_i.
2	Split input string into separate parts, each part holding exactly one symbol.
3	Feed all the single symbols one by one into the network by assigning their random codes to the sensor activation values. Repeat steps 4 through 8 for all elements of the training set.
4	Set input values on the sensors of the first layer and concatenate the symbol and recurrent vectors: $\underline{x}^{(1)}(t) = \begin{cases} \underline{x}^{(1)}{}_s(t) \oplus \underline{\varrho} \; first \\ \underline{x}^{(1)}{}_s(t) \oplus \underline{x}^{(1)}{}_r(t) \; else \end{cases}$ $\underline{w}^{(1)} = \underline{w}^{(1)}{}_s(t) \oplus \underline{w}^{(1)}{}_r(t)$
5	Calculate the average activation of the first map: $\underline{y}^{(1)}(t) = \frac{1}{(\underline{w}^{(1)}(t) - \underline{x}^{(1)}(t))^2 + \delta}$ $Y(t) = \sum_{i=1}^{N^{(1)}} (\underline{y}_i(t))^2$ $\underline{y}^{(1)}(0) = \underline{\varrho}$ if first element of string $\underline{y}^{(1)\prime}(t) = \frac{(\underline{y}^{(1)}(t))^2}{Y(t)}$ $\underline{y}^{(1)\prime\prime}(t) = \frac{\omega \cdot \underline{y}^{(1)\prime}(t) + (1-\omega) \cdot \underline{y}^{(1)\prime}(t-1)}{\omega}$

Step	Description
6	Reduce dimension $\underline{y}^{(1)''}(t)$ from $N^{(1)}$ to n_r Copy result into the activation sensors of the second map and into the recurrent fibres of the first map: $\underline{x}^{(2)}(t) = \sum_{i=1}^{N^{(1)}} (\underline{y}_i^{(1)''}(t) \cdot \underline{\zeta}_i)$ $\underline{x}_r^{(1)}(t+1) = \underline{x}^{(2)}(t)$
7	Determine minimum map L: neuron v. This neuron has the network's best match between the input values and its weight values: $v^{(L)}: \quad \forall k \left\| \underline{w}^{(L)}{}_v(t) - \underline{x}^{(L)}(t) \right\| \leq \left\| \underline{w}^{(L)}{}_k(t) - \underline{x}^{(L)}(t) \right\|$ for k element of the feature map L, L = 1,2...
8	Update all weights in the map according to the Kohonen training rule: $\underline{w}^{(L)}{}_k(t+1) = \underline{w}^{(L)}{}_k(t) + \varepsilon(t) \cdot \Phi_{kv} \cdot (\underline{x}^{(L)}(t) - \underline{w}^{(L)}{}_k(t))$ for k element of the feature map L, L = 1,2 $\Phi_{rs} = e^{-\|k-v\| \cdot 2\sigma^2(t)}$ $\varepsilon(t) = \varepsilon_{max} \cdot \left(\frac{\varepsilon_{min}}{\varepsilon_{max}}\right)^{\frac{t}{t_{max}}}$ $\sigma(t) = \sigma_{max} \cdot \left(\frac{\sigma_{min}}{\sigma_{max}}\right)^{\frac{t}{t_{max}}}$, where $\omega \in [0,1]$ Memory rate, $\varepsilon_{max} \in [0,1]$ Initial learning rate, $\varepsilon_{min} = [0,1]$ Final learning Rate, $\sigma_{max} \in [0,1]$ Initial region size, $\sigma_{min} \in [0,1]$ Final region size, $\|k-v\|$ Physical distance from neuron k to v

5.0 Experimental Results

All simulations were implemented on a Sun Sparc Station IPC in C. The simulator used sentences of three different input types to evaluate the model: sentence schemata of the form {NOUN VERB NOUN}, sentences of the form {John loves Mary} [Elman, 1988], and strings of the form {ABC} [Cleeremans et al., 1989].

It has been the aim of the simulations to compare the model as presented here with other models known in connectionist natural language processing, in particular with Elman's *SRN* and Ritter and Kohonen's *Semantotopic Map*. Therefor, the data that they used in the simulations of their models is also used here. The results are compared in a qualitative manner. There are no comparisons made to any known statistical learning algorithms that implement similar behavior such as Principle Component Analysis or Group Error Analysis or Canonical Discriminants Analysis [Devijver et al., 1982].

5.1 Simulation Parameters and Convergence

The parameter values used for the simulations were about the same for all three different types of input (variables specified after step 8 in the algorithm). However, different parameters were mainly influencing (temporal and element) memory capacity and convergence speed. The following model constants worked best: $\omega = 0.3333$, $\varepsilon_{max} = 0.80$, $\varepsilon_{min} = 0.05$, $\sigma_{max} = 4.5$, $\sigma_{min} = 0.5$. In the used models, the first map consisted of 8 by 9 neurons, and the second map of 4 by 4 neurons. These numbers were derived by experimental means.

After some initial simulations, the relation between the amount and norm of input and recurrent fibres seemed very important. If the norm of the symbolic part was larger than the recurrent one, an ordering based on symbolic grounds instead of contextual ones developed. If, on the other hand, the recurrent norm was larger than the symbolic, the entire map converged towards the first element encountered in the training sequence. Therefore it is very important that the norm as well as the number of fibres of the symbolic vector and the recurrent vector are equal. In this context, the dimension reduction of the input vector in the second map plays an important role, based on reasons other than complexity. Without this reduction, the norm of the recurrent fibres would be too small, resulting in an ordering based on the internal coding of the symbolic elements. Now, both vectors are constructed from the same random set. The convergence process seemed to be quite complex. This is understood if one realizes that there are in fact two non-linear processes influencing each other.

At first sight, convergence seemed based on good luck rather than on logical foundations. However, a number of parameters can influence the convergence process. First, there is the sharpening of $y^{(1)'}(t)$. If this process is iterated multiple times by dividing its square value with $Y(t)$, it represents the most activated neuron better and better, resulting in a

faster convergence of the first map. Secondly, a large ω means shorter memory, but a faster convergence with small sentences. If strings become larger, ω must definitely be decreased to stop the system from having only single-word left-context sensitivity. A reasonable heuristic for the value of ω is *1/(average sentence length)*. Furthermore, there are the values for ϵ and σ, which have their specific influence on the self-organizing process as described in [Ritter et al., 1988]. Although convergence can be guided, convergence times are large and show chaotic behavior, resulting in a process which is complex and hard to monitor. Last but not least, one must be aware of the fact that the organization of the second map (contextual structures) only starts after the first map is about to be organized. This implies that multi-level organization in models similar to the one proposed here, takes probably at least twice as long as in single layer models.

5.2 The Internal Coding of the Symbols

The internal coding of the symbols was generated and assigned randomly, although the internal distance was constant (e.g. only 0.00, 0.33, 0.66 and 1.00 were used as legitimate values by the random generator). In early simulations, completely random numbers were used. However, if these values happened to be too closely related, convergence speed could decrease significantly. Therefore, this more artificial coding scheme was chosen to implement the symbol codes. In speech recognition or computer vision applications, the Kohonen feature map has frequency, light intensity, and contrast ranges as input, all natural data input types. Here, one uses an artificial coding for words and sentences. Therefore, one can defend the choices made in applying this coding scheme. The final code for a symbol is assigned randomly. The generation of different codes, is based on permutations of the base values. As far as possible, the complete coding space was used, so element codes were distributed equally throughout the element space (if a certain part of the map had a higher clustering density, then this was based on frequencies of occurrence and not on characteristics of the internal coding). Some examples can be found in Table 1, "Internal coding of the elements".

TABLE 1. Internal coding of the elements

Nr.	Verb type	Dim. 1	Dim. 2	Dim. 2	Dim. 3	Dim. 4	Dim. 5	Dim. 6
0	NOUN-HUM	0.0000	0.0000	0.6667	1.0000	0.0000	0.6667	0.0000
1	VERB-EAT	0.0000	0.3333	0.3333	0.6667	0.3333	0.0000	0.0000
2	NOUN-FOOD	0.0000	0.6667	0.0000	0.3333	0.3333	0.6667	0.0000
3	VERB-PERCEPT	0.0000	0.6667	1.0000	0.0000	0.6667	0.0000	0.0000
4	NOUN-INANIM	0.0000	1.0000	0.3333	1.0000	0.6667	0.6667	0.0000
5	VERB-DESTROY	0.3333	0.0000	0.0000	0.6667	1.0000	0.0000	0.0000
6	NOUN-FRAG	0.3333	0.0000	1.0000	0.3333	1.0000	0.6667	0.0000
7	VERB-INTRAN	0.3333	0.3333	0.6667	0.3333	0.0000	0.0000	0.0000
8	VERB-TRAN	0.3333	0.6667	0.3333	0.0000	0.0000	0.6667	0.0000
9	VERB-AGPAT	0.3333	0.6667	1.0000	1.0000	0.3333	0.0000	0.0000
10	NOUN-ANIM	0.3333	1.0000	0.6667	0.6667	0.3333	0.6667	0.0000
11	NOUN-AGGRESS	0.6667	0.0000	0.3333	0.3333	0.6667	0.0000	0.0000

5.3 Type 1: Semantically Tagged Syntactic Structures

To start, there are 16 simple sentence schemata consisting of syntactic categories (see table: Type 1: Syntactic Sentence Structures.). This formalism was used to illustrate the formation of a topological map of syntactic equivalents. Next, ordinary words were substituted for the constituents, resulting in the final training sentences. These sentences show how the model derives syntactic classes from flat strings. Because these two input types provide no insight in the grammatical capabilities of the model, a third type (generated by a finite state machine) was introduced to evaluate the model's grammatical learning power.

TABLE 2. Type 1: Semantically tagged syntactic sentence structures

Nr.	String
0	NOUN-HUM VERB-EAT NOUN-FOOD
1	NOUN-HUM VERB-PERCEPT NOUN-INANIM
2	NOUN-HUM VERB-DESTROY NOUN-FRAG
3	NOUN-HUM VERB-INTRAN
4	NOUN-HUM VERB-TRAN NOUN-HUM
5	NOUN-HUM VERB-AGPAT NOUN-INANIM
6	NOUN-HUM VERB-AGPAT
7	NOUN-ANIM VERB-EAT NOUN-FOOD
8	NOUN-ANIM VERB-TRAN NOUN-ANIM
9	NOUN-ANIM VERB-AGPAT NOUN-INANIM
10	NOUN-ANIM VERB-AGPAT
11	NOUN-INANIM VERB-AGPAT
12	NOUN-AGGRESS VERB-DESTROY NOUN-FRAG
13	NOUN-AGGRESS VERB-EAT NOUN-HUM
14	NOUN-AGGRESS VERB-EAT NOUN-ANIM
15	NOUN-AGGRESS VERB-EAT NOUN-FOOD

5.4 Semantotopic Map of Syntactic Structures (Type 1 Input)

After 380,000 training cycles the following formation developed on the first map (upper part of First and Second Map After 380.000 ... Constituents). Every element represents a word fitting best in relation to the contents of the neuron. Two possible approaches can be taken here. Either one determines the best matching neuron for each symbolic object, or one determines the best symbolic objects given a vector code in the feature map. Here, the latter one has been used. XXXXX means no reasonable mapping could be found. One can clearly distinguish the NOUN from the VERB part on the first map. Interesting are the neighborhoods holding related syntactic-semantic objects like NOUN-ANIM/NOUN-INANIM, VERB-TRANS/VERB-INTRANS and VERB-DESTROY/NOUN-AGGRESS.

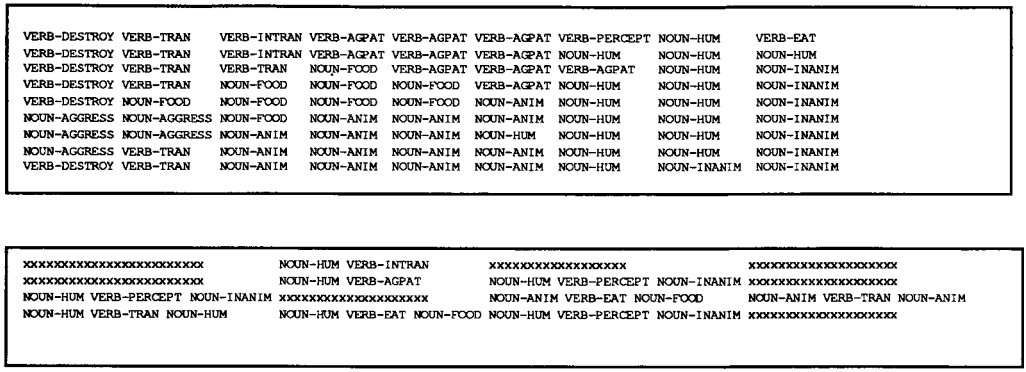


FIGURE 2. First and second map after 380.000 training cycles with syntactical categories

The second map (lower part of figure 3) represents related contextual structures. On the left are sentences starting with a NOUN-HUM, on the right are the ones starting with NOUN-ANIM. Sentences holding VERB-EAT concentrate respectively at the left and the right side of these regions, resulting in a simple syntactic generalization. So, the network classifies and generalizes on semantic features as well as on syntactic structures.

One must realize that the second map is not needed for context-sensitivity. A model without a second map (but with the recurrent fibres as described), processes sequential information (prediction of next element, grammatical correctness) just as well. However, the syntactic structures used in the disambiguation and structure assignment process do need information from the second map. The results of the prediction and grammar-checking process, improve by using additional information of the second map.

Even after so many cycles, the map still had not converged completely (to the state of self-organization from which there is no escape). However, results indicate that the network is converging towards such a state, which will probably be achieved after more than a million cycles. (In the current implementation, training the network with one million training cycles takes more than a week of calculations).

Prediction is possible in the model by feeding forward one element, determining the recurrent fibre and finding the best matches of this fibre on the first map. The symbolic parts of these neurons hold possible subsequent elements in the string. If the next element is predicted correctly, the string is accepted. Otherwise, the string would be rejected as being ungrammatical. By using a threshold in the matching process between the input and the

weight of the recurrent fibres, one can determine the measure of correctness, an interesting value which provides an indication of the quality of the prediction. After training the following results were obtained:

TABLE 3. Some string prediction examples

Nr.	String	Current string element	Prediction of next string element
0	NOUN-HUM VERB-INTRAN	NOUN-HUM:	{VERB-INTRAN or VERB-AGPAT}
		VERB-INTRAN:	{}
1	NOUN-HUM VERB-AGPAT NOUN-INANIM	NOUN-HUM:	{VERB-TRAN or VERB-AGPAT}
		VERB-AGPAT:	{NOUN-INANIM}
		NOUN-INANIM:	{}

5.5 Type 2: Sentences Generated From Syntactic Structures and Word Class Substitutes

In the first experiment sentence schemata from type 1 were taught to the network. The examples shown in Table 2, "Type 1: Semantically tagged syntactic sentence structures" are the only ones learned (in random order, multiple times). The second experiment used examples of type 2, which were generated from type 1 syntactical structures and word class substitutes. The algorithm randomly selects words for each class and substitutes them in the syntactical framework. Although simple, the sentences produced are complex enough to demonstrate the self-organizing capabilities of the model.

TABLE 4. Word class substitutes

Nr	Verb Type	Substitution
0	NOUN-HUM	man, woman
1	NOUN-ANIM	cat, mouse
2	NOUN-INANIM	book, rock
3	NOUN-AGRESS	dragon, monster
4	NOUN-FRAG	glass, plate
5	NOUN-FOOD	cookie, bread
6	VERB-INTRAN	think, sleep
7	VERB-TRAN	see, chase
8	VERB-AGPAT	move, break
9	VERB-PERCEPT	smell, see

Nr	Verb Type	Substitution
0	NOUN-HUM	man, woman
10	VERB-DESTROY	break, smash
11	VERB-EAT	eat

The result shown in Table 5, "Type 2: Sentences generated from syntactic structures and word class substitutes" is a subset of the 100.000 sentences generated. The first number indicates the number selected from 100.000 sentences, the second number indicates the training cycle number.

TABLE 5. Type 2: Sentences generated from syntactic structures and word class substitutes

Training Nr.	String
0	Jim works seldomly
1	Jim eats beer
2	Jim walks slowly
3	dog drinks water
4	Mary hates bread
5	Bob drinks seldomly
6	Bob runs well
7	cat walks seldomly
8	Jim eats water
9	Jim likes cat
10	Bob hates water
11	Mary hates water
12	horse runs poorly
13	cat likes bread
24	cat likes cat
25	dog eats water
26	Jim works slowly
27	Jim sells slowly
28	dog eats seldomly
29	Bob speaks slowly
30	cat eats slowly
31	Jim speaks poorly
32	Mary hates bread
33	Mary drinks seldom
34	cat eats seldom
35	Mary hates dog
36	cat walks well
37	dog eats water

5.6 Semantotopic Map from Unformatted Sentences of Type 2

The following table presents an example of the formation of a semantotopic map of sentences of the type {John loves Mary} after 50,000 iterations. The map was formed from unformatted (flat) sentences passing by. The syntactic (or low-level semantic) category as well as the relation between categories can be derived from the position on the map.

man	man	monster	xxxxxxx	dragon	xxxxxx	break
woman	move	smash	see	smell	eat	chase
break	glass	smash	smell	break	break	chase
break	break	glass	cookie	break	bread	xxxxx
break	break	break	cookie	mouse	bread	xxxxx
plate	glass	break	mouse	cat	cat	book
plate	xxxxx	glass	break	mouse	rock	book

FIGURE 3. Feature map after 50.000 training cycles

Although there is not yet a clear separation on the map between nouns and verbs, the map holds substitutionally related elements in neighboring regions. If similar maps are studied at vast time intervals, the map seems to alter periodically between convergence and divergence, shaking objects on the map, which results in a better organization after a while. This type of behavior is characteristic for Kohonen feature maps. Here too, better results can be expected after more than a million training cycles.

During the training sessions, the number of times a neuron was assigned Best Matching Unit (BMU) was counted. Peaks in this distribution are spread over the map, showing the balance between recurrent and symbolic fibres. The high number 895 in the left corner is caused by the fact that the recurrent fibres of the first element in a string is set to zero, triggering an increased activation of this neuron. Here, only sentences of three words were used, resulting in relatively many sentence starts. If one counts these activations in the FSM simulations, two peaks are found, one for the start and one for the end symbol. As sentences have a more varied length, these frequencies are more naturally distributed.

175	51	112	53	315	37	268
70	100	72	77	58	5	165
116	77	49	86	119	6	52
67	28	56	62	78	24	66
62	7	129	38	64	82	130
118	76	76	15	15	251	15
214	376	24	420	11	4	895

FIGURE 4. Activation frequency of the elements in the map

5.7 Type 3: Strings Generated by a Finite State Machine (FSM)

Type 3 sentences are used to test the system's ability to predict and recognize elements of strings generated by a Finite State Machine (FSM) [Cleeremans et al., 1989] (see figure 5, "A finite state machine (FSM)").

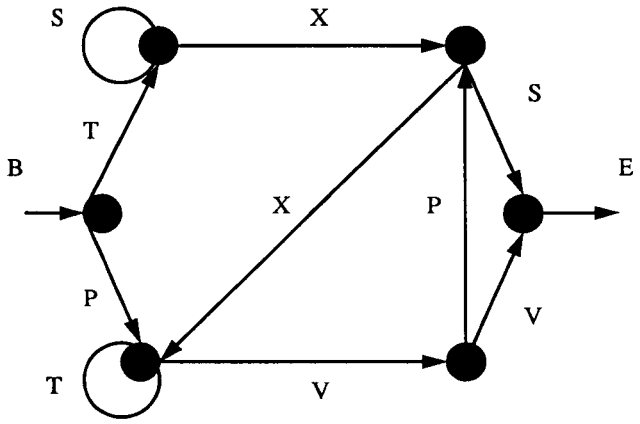


FIGURE 5. A finite state machine (FSM)

TABLE 6. Type 3: Strings generated from the FSM

Selection Nr.	Training Nr.	String
2581	0	bpttvve
692	1	btxxvve
5285	2	bpvpse
676	3	btxxtvve
5387	4	btsssxxtvpxvve
7592	5	bpvpse
9899	6	bpvpse
4372	7	btxse
5525	8	bpttvve
3281	9	bptvpse
2086	10	bpvve
7158	11	bptttvpse
7117	12	bpvve
9719	13	bpvpse
176	14	bpvpxtvpse

Selection Nr.	Training Nr.	String
8374	15	bpvpse
8611	16	bpvpxtvpxtvve
15	17	btxxttvve
793	18	bptvve
6454	19	btsssxxttvve

If a state results in two directions (such as the first state, where one can choose between a T and a P), one is chosen randomly with a probability distribution of 0.5. The strings shown form a subset of more than 1 million generated (See Table 6, "Type 3: Strings generated from the FSM"). The FSM has some characteristics making it interesting for simulations. First, there are multiple choices in every state. Secondly, the recurrent transitions with S and T can result in long sequences, testing the network's memory capacity. Third, all elements occur at different transition places, forcing the system to remember (and use) a lot of context to determine grammatical correctness or to predict the next element in a string.

To determine a grammatical lower bound for the model, a Finite State Machine (FSM) similar to the one in [Cleeremans et al., 1989] was implemented to generate simple sentences. Although one cannot prove the theoretical equivalence between a recurrent neural network (RNN) and a FSM, experiments can indicate there is one. If for any Finite State Language (FSL) the network can be trained to accept all strings to this language and to reject all others, then the RNN probably implements at least a FSM: a grammatical lowerbound. This lowerbound is determined experimentally and has no mathematical value whatsoever.

5.8 Grammatical Correctness

More than 1 million sequences were taught to the network. Afterwards, the network was tested with randomly generated sentences of which only 10% was grammatical. 99% of all grammatical strings were accepted, 1% rejected. Of all ungrammatical strings 0% was accepted. The main reason for the 1% failure of the grammatical strings was the long repetitions of *tttt* and *ssssss*. Even after a million cycles one could define a sentence which would not be recognized (e.g. a sentence containing more than hundred t's after each other). However, the longer the training process, the better the performance. Below some examples of the simulations are shown (see Table 7, "Some string acceptance and rejection examples"). Here too, a threshold function was used. If the error of the matches between the input values and the weights of the recurrent fibres grew above this value, the string was rejected; if the end of string symbol was reached before the threshold, the sentence was accepted. During the simulations, two different threshold models were used:

one which compared the cumulative difference to a threshold, and one which compared the difference between input and weight values of every character to a threshold. The last model worked much better than the first one.

TABLE 7. Some string acceptance and rejection examples

Nr.	String	Accepted	Grammatical
0	btxse	YES	YES
1	bpttttttvve	YES	YES
2	bpvve	YES	YES
3	bse	NO	NO
4	bxe	NO	NO

5.9 Disambiguation

Another linguistic task is word-sense disambiguation. Although hard to imagine, an organization can be achieved where all elements are relatively in such a position, that a statistical distribution is implemented which provides information on the plausibility of a certain meaning or structural function of a word, derivable from the regional part it activates. E.g., suppose one word has multiple meanings. This results in the formation of just one region on the map. However, this region is constructed of sub-regions for each different meaning of the word. If an additional shell keeps track of the position of words on the map, the exact meaning of a word within a context can be determined from the sub-region which is activated. If a structure has different meanings, the same algorithm can be applied to the second map, where a structure region is constructed out of ambiguous sub-structure regions.

In this context it is important to realize that as long as the words are ordered relatively to each other, as many different relations as possible can be stored in a feature map. There is more than a two- dimensional ordering. As a matter of fact, this ordering is a projection of the multi-dimensional space where each dimension of the input vector (and weight vector) represents a dimension in that space. If one sees the ordering of words in this context, one might imagine a relative ordering capable of a disambiguation task. The capability of the network to indicate a measure of correctness plays a particularly significant role in this process. By calculating the difference between the input values and the weight values, one can determine the part of the region that corresponds best to the input values. The additional shell mentioned above would assign a meaning to the region indicated, thus disambiguating the word.

The disambiguation task has only been investigated globally. The current model (simulation) has some disadvantages causing problems which can be solved better after some changes to the model as a whole. Most important is the insight about how one can disam-

biguate with a Kohonen feature map. The same holds for the linguistic task described in the next paragraph: assigning structures to constituents.

5.10 Assigning Structures to Constituents

Even more ambiguous is the task of assigning structures to constituents. The second map derives a distribution of the structures of the sentences presented to the first map in an unsupervised way. If an additional shell keeps track of the position of these regions after (and during) the processing of such sentences, it is possible to assign a structure to a sentence which is fed forward through the network. However, before the sentence structures are derived from scratch, a very long training process must be undergone. Besides this disadvantage, one has to be aware that structures assigned to sentences are based on a statistical distribution of left- context sensitive words sequences. So, complex structures might not be noticed, or be confused with others. Mainly due to the long training cycles, this task has not been evaluated in depth: only some simple (but successful) tests have been made.

On the one hand, the statistical characteristics of the distribution can be helpful, on the other, the inexact match between a sentence presented and a structure on the map, can cause confusion. Once more, all this is based on the cooperation with an additional shell which knows exactly the locations of word meanings and sentence structures on the map. The feature map indicates the most plausible solution to a problem by activating the region holding the distributed information, but the final structure is assigned by the shell. It is impossible to store the explicit structure (E.g. Noun-Verb-Noun) in the feature map. But who cares about these artificial notions anyway?

5.11 Overall Results

The overall results of the simulations are good. Although some remarks can be made such as:

- Very long training times
- Unstable character of the convergence process
- Lack of real insight in the convergence process, resulting in a situation where one never knows whether the map has finished converging, or if a local minimum has been reached

On the other hand, the model seemed to be remarkably good in performing various linguistic tasks (although simulated in sometimes very simple ways) such as:

- Word classification
- Word class derivation
- Sentence structure derivation
- String prediction
- Grammar checking
- Disambiguation
- Structure assigning

Furthermore, the model worked in cooperation with a traditional shell which preprocessed the information and took care of the outputs generated by the model. This cooperation is not based on a synthesis between different mechanism within one model (such as a neural network extended with a stack), but on two different mechanisms incorporated in such a way that both do what they are best at. Therefore, the results are interesting enough to continue the research in the direction taken: self-organizing (unsupervised) temporal-processing models in natural-language processing inspired by neurological models of brain structures.

6.0 Discussion

By combining and re-interpreting work of [Ritter et al., 1989b], [Ritter et al., 1990], [Kangas, 1990] and [Elman, 1988] a model was developed which can derive a semantotopic map of language from unformatted strings. In this section, some important aspects of the model are discussed.

6.1 Grammatical Processing Capabilities

After theoretical analyses of the model, simulations based on examples from Elman's Simple Recurrent Network (SRN) [Elman, 1988], [Servan-Schreiber et al., 1988, 1989, 1991], [Cleeremans et al., 1989], resulted in similar results as obtained in their research. The results obtained during these simulations are relevant because they were based on exactly the same data as used in the references above. So, in this case, it was possible to compare the Recurrent Kohonen Maps with the SRN and the Semantotopic Map.

The model predicts new elements in a sequence, categorizes words according to their syntactical and semantical features, accepts finite-state grammars, derives contextual structures, categorizes these structures and generalizes over the information stored in both layers. Moreover, all features were learned by using a unsupervised training algorithm. The first map categorizes single words, the second map derives and categorizes contexts. A limitation of the model appears in the task of recognizing very long sequences (as indicated in the previous section). This lack in performance can be overcome by increasing training-times. However, this research could not determine an upper bound.

Which grammar classes are recognizable by the network type is very important in this context. In [Tsung et al., 1989] and [Cottrell et al., 1989, 1990] it was shown that context feedback networks (Elman networks) could perform tasks which could not be solved by state feedback networks (Jordan networks). Next, [Cleeremans et al., 1989] proved that Finite State Grammars (FSG) are a lower bound for Elman networks. Recurrent fibres only cause a left-context sensitivity. More complex grammars such as context sensitive grammars are not possible without introducing right-context sensitivity, which might only be possible by addition of buffering mechanisms (or short term memories). Related research determining the grammatical capacities of recurrent networks can be found in [Giles et al., 1990, 1991], [Liu et al., 1990], [Sun et al., 1990a-b] and [Chen et al., 1991] where context-free languages are recognized by using higher order networks (recurrent fibres and state memories).

Here, it was experimentally shown that recurrent Kohonen feature maps at least implement finite state behavior by training the model with strings generated by such a machine. All such strings were recognized by the model. In addition, the model never accepted a string not generated by the finite state machine.

6.2 Convergence Properties

Although the model seems to converge, this process is very complex and difficult to monitor or influence. By decreasing the number of recurrent fibres, complexity decreases somewhat. Generally, there are two methods for performing this task: first one can reduce the dimension even more, second, instead of using fully interconnected layers, one might use Gaussian connections (not all neurons between the layers are interconnected, just a few are. The probability of a neuron being interconnected is normally distributed, also indicated by the term Gaussian). The latter has the advantage of eliminating noise from irrelevant fibres (caused by their absence), which are still being used (although compressed) in the first option.

The balance between the number of recurrent and symbolic fibres is much too delicate in this model. If the norm or number of sensors is changed, convergence might not take place any longer. This instability can be seen as a basic shortcoming of the model.

For the moment, it is not clear when the model converges and when it does not. There are factors such as the decreasing learning rate and region size with obvious influences on the process. Furthermore, one can deduct the strict relation between the number of symbolic and recurrent fibres. But real insight into the process is lacking. In the single layer Kohonen feature maps, more is known on the convergence properties [Ritter et al., 1986], [Ritter et al., 1988], [Ritter et al, 1989a] and [Ritter, 1989].

In the model proposed here, two self-organizing models are influencing each other, resulting in a very complex process. It is useful to analyse this process in detail by mathematical means. However, research in this direction has not been carried out yet. See for instance [Simard et al., 1988], [Williams et al., 1988, 1989a-b] and [Pineda, 1987], where in depth analysis of recurrent back-propagation is presented. The same analysis (be it in less detail and with more restrictions) must be possible for recurrent self-organization. Initial tips for the mathematical framework of this research can be found in [Sontag, 1990] and [Tanaka, 1990]

6.3 Temporal Processing

One of the main issues of the research carried out happened to be the implementation of temporal processing capabilities in (self-organizing) neural networks. In the introduction, it was mentioned that there are globally four different directions: dimension extension, windows, buffering and recurrent fibres. The more current research proceeded, the more recurrent fibres seem to lack the power needed for natural language processing. Of course, recurrent fibres are very important as a feed-back function [Dell, 1985], but they are not powerful enough to take care of all temporal processes needed in a NLP system. Where the dimension extension and window mechanisms are not really plausible, flexible or ele-

gant, the buffering mechanisms as proposed by [Kohonen et al., 1981] and indicated in [Miller, 1956] show some additional advantages. However, the main problem is the implementation of such a system in a neural network without destroying the neural network computing paradigm and designing solutions which are very interesting from an engineering point of view, but not really from a linguistic or psychological one. Therefore, main efforts in future research shall be directed towards the development of more powerful temporal processing mechanisms.

6.4 Topological Maps of Language and Cognitive Maps

As mentioned, this model does not fit in the context of natural language processing at first sight. How does one, for instance, define a topological map of language? Or, how must one add contextual information in order to avoid organization on grounds of arithmetic features of the internal coding?

The question as stated on the interpretation of a topological map of language can be seen in light of the task and performance of the two maps: the unsupervised syntactical and semantical derivation and categorization of elements and structures from bare sentences just passing by. An even more interesting way of looking at this question is in the use of the feature map in the disambiguation task. As a two-dimensional projection of a multiple dimensional input space, the Kohonen feature map forms a relative distribution of various word senses and sentence structures. This feature map can be compared with a topological map of language, where objects that are closely related (according to some features) have smaller distances to each other than to less related objects.

Most NLP models use the Kohonen map in cooperation with a back-propagation algorithm as an optimum clustering device. Here, it is shown that Kohonen models are capable of processing symbolic data even in tasks other than conceptualizing. The question raised earlier on the interpretation of a topological map of language might, in the light of the effects on the second map, be answered by comparing its function with a part of the cognitive maps described in [Stolcke, 1990] and elsewhere in the literature [Lakoff, 1988] [Chrisley, 1990].

6.5 Related Work

Although the Kohonen self-organizing model is just an efficient statistical classifier, it is capable of deriving semantical features of symbolic data, as long as data is presented in its proper context. The same feature of neural networks can be seen in work carried out by [Miikkulainen et al., 1988a-b], [St. John et al., 1988a-b, 1990], where generalization over context resulted in the automatic derivation of semantic (micro) features. This ability of neural networks in is not found in classical symbolic AI.

Recent work done in recurrent or spatio-temporal self-organization shows a lot of variability in theory, architecture and implementation [Kangas, 1990], [Kangas et al., 1990], [Koikkalainen et al., 1990], [Samaranbunda et al., 1990], [Silverman, 1988], [Stotzka et al., 1990], [Tavan et al., 1990], [Thacker et al., 1990], [Yen et al., 1990]. Other work by Linsker and the even more biologically inspired Neuronal Group Selection theory of Reeke & Edelman might also be suited to implement linguistic phenomena [Edelman, 1987], [Reeke et al., 1988] [Reeke et al., 1990].

The main problem with all these unsupervised models is the complexity of the simulations and the less developed foundations, making NLP application research quite tricky. Various hybrid solutions try to overcome this disadvantage of self-organizing models. A possible solution is to use a self-organizing feature map to discover the features in the training set, and back-propagate between these maps to train and generalize between input and output pairs (or between input patterns and regions on the map). The efficient back-propagation algorithm then limits the complexity and uses known mechanisms, like recurrent connections, to implement complex phenomena.

More on these solutions can be found in [Hrycej et al., 1989, 1990] and [Gersho et al., 1990]. Other hybrid solutions where either self-organizing and back-propagating networks, or self-organizing networks and symbolic techniques or back-propagation neural networks and symbolic methods are combined, are described in [Dolan et al., 1987], [Dyer, 1988], [Dyer, 1990a-b], [Honavar et al., 1989, 1990], [Jain et al., 1990].

6.6 Future Work

Future research may concentrate on the implementation of other linguistic tasks such as disambiguation, parsing, language acquisition and the derivation of simple analogies. Besides these applications, extensions to the model are being implemented based on a stronger synthesis between cognitive science and neuroscience. Gaussian interconnections, additional layers and more Hebbian training rules are an indication of the direction chosen. Furthermore, other mechanisms of processing temporal information must be developed which are more powerful than recurrent connections as used in this research.

A more basic shortcoming to the results presented here (and that of the entire connectionist natural language processing community) is a more quantitative understanding of the relation between Recurrent Kohonen feature maps and traditional statistical learning algorithms. There has been some research towards comparison of the SRN with some statistical methods. However, there are no quantitative results known yet [Dennis et al., 1991]. There has been more research towards the comparison of Kohonen feature maps and Hidden Markov Models (HMM) for speech recognition [Bourlard et al., 1989, 1991]. It would be an interesting task to implement these comparisons for the simulations as presented here.

Lately, this relation between self-organizing feature maps (which are computationally efficient with respect to the fact that they are self-organizing) and more biological models which are based on neuroscientific brain research, but are very hard to simulate with computer implementations, are getting more and more attention. By extending the Kohonen formalism towards multi-layer models with some variations in neuron type and training rules, one can use an already evaluated theory in new models. Experiments in this direction are found in [Erdi, 1990] and [Ojemann, 1983].

So, future work may involve model extensions and developments in the spirit of the Neuronal Group Selection theory [Edelman, 1987] or inspired by a stronger synthesis between cognitive science and neuroscience [Eimas et al., 1990]. In addition, a more fundamental model for (connectionist) language acquisition [Weber et al., 1990] [Feldman et al., 1990a-b] can be used to evaluate acquisition performance in other connectionist models for NLP.

Chapter 4

Filtering the Pravda with a Self-Organizing Neural Network

"The best information retrieval systems are those systems that provide new answers to new questions"

Abstract

This chapter presents **two** types of implemented neural methods for free-text data base search. In the first **method, a** specific interest (or "query") is taught to a Kohonen feature map. Next, large amounts of unstructured text are passed along the network. Depending on the activity patterns that occur on the network, a text can be selected by the system. Here, the query is fixed and the data base is the variable part of the model. The second method works the other way around. Statistical properties (n-gram or keyword distributions) from various texts are taught to a feature map. A comparison of a query with this feature map results in the selection of texts closely related to each other with respect to their contents. So, here the data base is fixed and the query is the variable part of the model.

Various simulations show that for both networks, the neural network indeed converges towards a proper representation of the objects that it is taught. The first algorithm seems well scalable (linear time and memory complexity), resulting in high speeds, few memory needs, and easy maintainability. The second one particularly shows an elegant and uniform generalization and association method, increasing the selection quality.

By combining research results from connectionist Natural Language Processing (NLP) and Information Retrieval (IR), a better understanding of neural networks in NLP and an increased Information Retrieval quality are obtained.

1.0 Background

The Information Retrieval (IR) problem has many facets. The queries as well as the data base elements may be by either static or dynamic. Information filtering pertains to static queries in a dynamic data base environment. Here, one teaches a specific interest to a filtering device, which selects interesting text with respect to this interest. Regular free-text search refers to a more static data base with changing queries. Due to its static character, the data base can be preprocessed. In the retrieval phase, one compares the statistic analysis of a query with all the analyses of elements in the data base. Highly correlated analyses suggest a common topic [Croft et al., 1979].

The method of analysis in IR varies between statistical pattern recognition and a symbolic linguistic approach. Clearly, the retrieval quality depends heavily on the amount of context and conceptual knowledge that is available in the retrieval phase. However, linguistic approaches result in complicated and computationally complex systems that are not quite relevant in practical implementations. On the other hand, statistical pattern recognition techniques are quite unable to handle conceptual relations and higher order grammatical inferences, which are important to get the retrieval quality above the level of global surface analyses. Generally, IR systems use statistical matching methods on either characters or words and the analysis of meaning does not go beyond the use of synonyms [Rijsbergen, 1979], [Lancaster, 1979], [Salton, 1968, 1971, 1980b, 1986, 1989] [1].

Research in neural networks shows good results in various pattern recognition tasks. Implicit parallelism, easy incorporation of knowledge from different sources, good generalization and easy association capabilities are the well known examples of advantages of neural networks. So why not use them for yet another classification task: Information Retrieval.

In other words: *can one add more (relevant) context to IR by using a neural network?*

Recent research in connectionist Natural Language Processing (NLP) showed interesting results in self-organizing systems [Elman, 1988], [Scholtes, 1991a-i]. Other research shows automatic categorizations of unknown words into clusters which might be used to incorporate a simple notion of meaning in IR [Ritter et al., 1989b, 1990], [Elman, 1988], [Scholtes, 1991a-i]. Although these methods are not capable of analyzing complex linguistic structures, they do distinguish different contents better than global surface analyses, while they are still based on automatically derivable training and retrieval algorithms. (And if implemented on parallel hardware, these algorithms would also be fast).

1. See the discussion of this Chapter for a definition of the neural models in traditional Information Retrieval notions.

The research reported here describes the development of a Neural Filtering mechanism for (dynamic) free-text data bases. The neural (Kohonen) feature map shows considerable success in scalability, implementation ease, generalization and selection power. If implemented on parallel hardware, the neural method definitely outperforms any known text matching algorithm which is based on adjacent character or word statistics.

If the data base is a more static one, objects can be clustered by (predetermined) keywords, abstracts or even by the full text. For any specific query, one can then determine the best group instead of the best paper. In neural networks, such clusters can be formed on feature maps. If the neuron that correlates best with the query is found, the papers represented by this neuron -and by the neighboring ones- are probably related to the query.

The last model is more suited for static data and dynamic queries, where the filter fits better with static queries and dynamic data.

On the one hand, connectionist NLP techniques can increase the retrieval quality. On the other hand, the IR problem can contribute to the understanding of neural networks as pattern classifiers by comparing neural information retrieval with (already well known) statistical information retrieval results.

Parts of this work have been published in [Scholtes, 1991j-m, 1992a-d] and [Scholtes et al., 1992a-b].

2.0 Introduction

In this chapter, two models shall be presented:

- A Neural Filter
- A Neural Interest Map

The neural filter implements a mechanism in which a (large) query or interest description, stated in natural language, is taught to a self-organizing neural network, which derives an internal representation of the text. This text is then matched against a large stream of incoming data. In short, the query is stored in a feature map, the data base is matched against this query. In a practical implementation of this model, multiple queries can be matched simultaneously (see Query A to D in Figure 1).

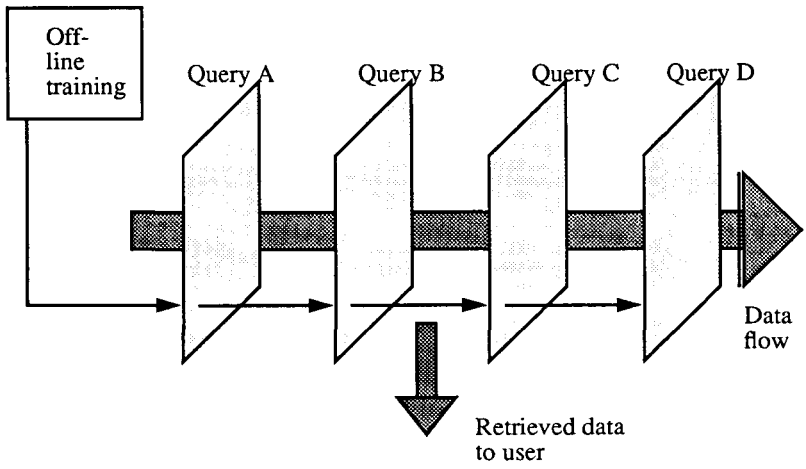


FIGURE 1. The neural filter principle

The neural interest map derives a full-text mapping from the documents in the data base onto a self-organizing feature map. Related documents are stored in neighboring areas in the neural network. A query is formulated by the user and matched against the data in the neural network. All documents represented by neurons within a certain distance from the best matching unit are retrieved and presented to the user.

The neural interest map constitutes the more obvious application of Kohonen feature maps in Information Retrieval. However, it shall be shown that the neural filter is much more suited for the application of neural techniques than the neural interest map is. This is partly

due to the lack of memory in the Neural Interest Map and partly to the standard problems that occur in clustering large document spaces.

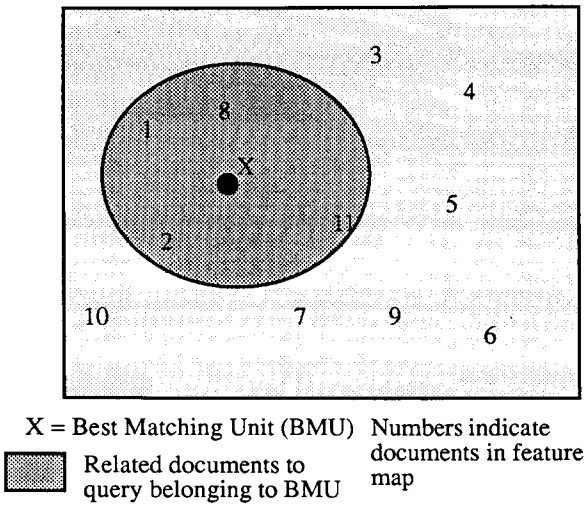


FIGURE 2. The neural interest map principle

Both models shall be implemented in a Kohonen Feature Map by using statistics about the adjacency of elements in the underlying text.

A statistical algorithm that incorporates such adjacency information is the n-gram vector method. An n-gram is an n-length sequence of characters occurring in a word. For example, the trigrams ($n = 3$) occurring in the word trigram are *--t, -tr, tri, rig, igr, gra, ram, am-, m--* (the - indicates a space). An n-gram frequency vector (which is equivalent to an n^{th} order Markov chain over characters) can be viewed as a document finger print; documents can be identified by such vectors. Normally, bigrams (2-grams) are not distinguishing enough, trigrams (3-grams) yield enough distinction and can be practically calculated, 4-grams do not add enough difference in feature vectors to justify the computational power, 5-grams are almost impossible to calculate and resemble keyword vectors. N-gram vectors provide enough distinguishing power only if common words and common endings are eliminated from the text trained to the neural map. Furthermore, by multiplying n-gram frequencies with weight values (high values for rare n-grams and low values for frequent n-grams), less frequent n-grams may be accentuated.

However, there are some major drawbacks.

- First, there is the Markovian nature of the model. It cannot remember strings longer than the order of the Markov chain, even when a larger context is relevant to distinguishing two objects. One can extend the order of the chain, but every step results in a rapidly increasing search space. So, the n-gram method is not really scalable to higher order dependencies (e.g., 5-gram character or word chains).
- Second, the implementation of higher order n-grams requires sophisticated programming techniques. The statistical tables must be hashed, ordered, and normalized. [1]
- Third, there is no meaning involved in the analysis method; only structural features of the text are taken in account.

Nevertheless, n-gram vectors are very powerful, easily manipulable, easy trainable and language independent [2] [Forney, 1973], [Hanson et al., 1974], [Neuhoff, 1975], [Shinghal et al., 1979a-b], [Hull et al., 1982], [Srihari et al., 1983, 1985], [D'Amore et al., 1988], [Kimbrell, 1988].

1. The many ways to optimize the statistical methods are known to the author. Much has been written on the string matching problem, clustering algorithms for n-gram vectors, etc., all resulting in better solutions for the brute force methods as proposed here. On the other hand, one does not incorporate these methods in the comparison because neural networks can be optimized in the same way with the same results [Kelly, 1991], [Koikkalainen et al., 1990]. Therefore, only plain, non-optimized methods are compared.

2. Although the quality of retrieval is increased by eliminating specific words and word-endings (which are in fact language dependent), this method is still categorized as being language independent because this is just a simple (very trivial) table of words. Normally, this list is not longer than 250 words and about 20 endings (see also the next footnote).

3.0 Models and Algorithms

This paragraph discusses the models and corresponding algorithms in detail. The models are based on the Kohonen training rule and on extensions of this model.

3.1 Algorithm 1.1: The Kohonen Neural Filter Based on Characters

The n-gram analysis method can be interpreted as a window size *n*, shifting over the words. This can be implemented quite simply in the Kohonen input sensors by assigning several sensors to each element in the window and concatenating all the window sensors to one big input vector. By shifting this window over the training text, only frequent n-grams form clusters on the feature map, the others are overruled (see figure 3).

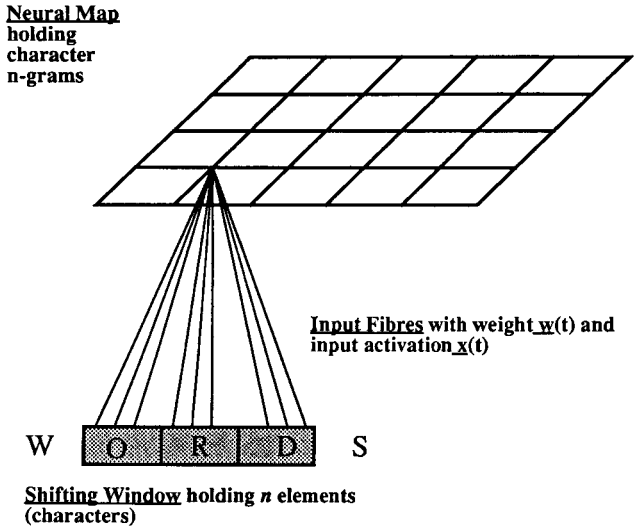


FIGURE 3. The neural filter based on character n-grams

After training, texts corresponding best to the query in the feature map will fit best to the clusters in the map (i.e., will yield the lowest cumulative error[1] with respect to the input values). Thus, this type of feature map can be used as a filtering device in an environment with a static query and a dynamic information flow. The method can be extended by incorporating spaces, so the model also learns adjacency relations between words. Such n-gram frequencies are often correlated with syntactic, semantic and sometimes even pragmatic information (see algorithm in Table 1, "Algorithm 1.1: Neural filter with n-grams").

1. The error between an input signal x and a weight vector w is calculated by $\sum_{i=1}^{N} \| w_s(t) - x_i(i) \|$, where s is the BMU on the feature map. More on this can be found later.

To reduce the noise and to restrict the input space, some measure must be taken. First, all lower case characters are transformed to upper case. Furthermore, all non-alphabetic characters are eliminated (digits, point, commas, etc.).

For reasons of efficiency, all irrelevant n-grams have to be eliminated so the rare ones are accentuated. Therefore, one has to remove non-relevant words within a language (e.g., *the, a, an, all, every, who, which,* etc.). This might sound awful to a psycholinguist, but one should remember that this method treats information filtering as a pattern recognition problem. Next, eliminate all common word endings such as: *-ing, -ant, -end,* etc. The remaining n-grams can be taught to the feature map in order of appearance according to the Kohonen formalism[1].

TABLE 1. Algorithm 1.1: Neural filter with n-grams

	Step A: Teach Query to a Neural Network
0	Initialize and determine best training parameters
1	Change lower case characters to upper case
2	Eliminate non-alphabetic characters
3	Eliminate non-relevant words
4	Eliminate non-relevant word endings
5	Determine n-grams
6	Teach n-grams to Kohonen feature map
	Step B: Pass Free Text along Neural Filter
0	Determine start and end of texts (line, paragraph, separator, etc.)
1	Change lower case characters to upper case
2	Eliminate non-alphabetic characters
3	Eliminate non-relevant words
4	Eliminate non-relevant word endings
5	Determine n-grams
6	For each n-gram: input n-gram to neural network and determine error
7	Select text if cumulative error < threshold

1. If enough data for a specific field is available, the detection of uniformly frequent (and therefore non-relevant) words can also be done automatically by a preprocessor. But, every language also has its own non-relevant words, which are applicable to all different corpora. The perfect filter set would contain the domain dependent as well as the domain-independent word set.

3.2 Algorithm 1.2: The neural filter based on words

In the second neural filtering algorithm, the system has access to a small dictionary of 500 to 1,000 words. Every word has an unique code of some sensor values. After elimination of non-relevant words (words that are not in the lookup table) and word-endings, a vector representing a Markov chain over words is calculated. This vector is taught to the system. After passing the training text multiple times, the Kohonen feature map contains a representation of common word combinations in the training text (see figure 4).

By processing the retrieval text similarly, the retrieval algorithm incorporates contextual relations. The measure of correlation between these vectors and the representation on the feature map, determines whether a text part can be selected or not. In this example all words are taught to the network. However, sometimes a word does not occur in the dictionary (because it is irrelevant for the selection process). The model ignores these words. As a result, it determines context from the relations between the remaining words (See algorithm in Table 2, "Algorithm 1.2: Neural filter based on word n-grams").

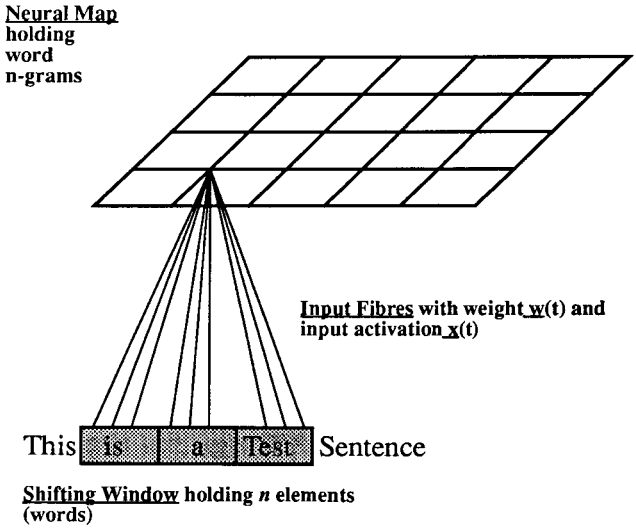


FIGURE 4. The neural filter based on word n-grams

TABLE 2. Algorithm 1.2: Neural filter based on word n-grams

	Step A: Teach Query to a Neural Network
0	Initialize and determine best training parameters
1	Change lower case characters to upper case
2	Eliminate non-alphabetic characters
3	Eliminate non-relevant words
4	Eliminate non-relevant word endings
5	Determine vector representing Markov chain on known words
6	Teach word n-grams to Kohonen feature map
	Step B: Pass Free Text along Neural Filter
0	Determine start and end of texts (line, paragraph, separator, etc.)
1	Change lower case characters to upper case
2	Eliminate non-alphabetic characters
3	Eliminate non-relevant words
4	Eliminate non-relevant word endings
5	For every text: for every word n-gram: determine vector representing word n-gram
6	Input this vector to the neural network and determine error
7	Select text if cumulative error < threshold

3.3 Algorithm 1.3: The Neural Filter Based on Dynamically Chosen Character N-Grams

Sometimes the vocabulary of a specific domain is not known exhaustively or it is very dynamic. In such a case the model first calculates the most frequent n-grams (for large *n* it approximates the average word length in a language). Then these n-grams are considered non-relevant, and they are eliminated from the training text.

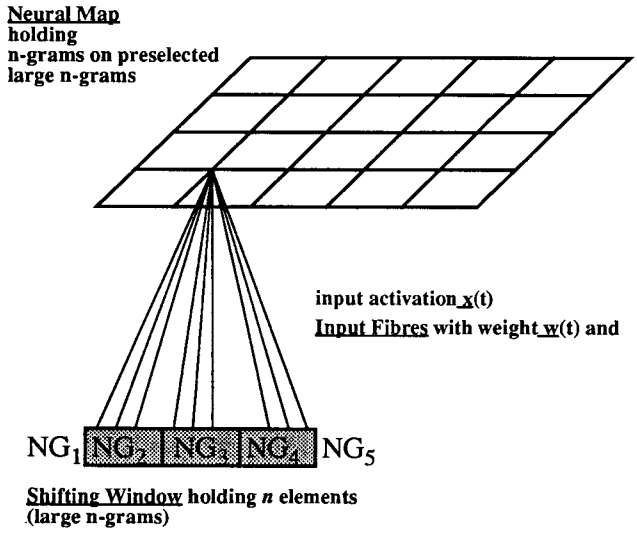


FIGURE 5. The neural filter based on dynamically chosen character n-grams

By shifting a window over the remaining n-grams, the neural map derives a representation of these n-gram combinations.

If the number of n-grams exceeds the addressing space, more n-grams may be eliminated from the training text manually or on the basis of frequency (See Table 3, "Algorithm 1.3: The neural filter based on dynamically chosen character n-grams").

TABLE 3. Algorithm 1.3: The neural filter based on dynamically chosen character n-grams

	Step A: Teach Query to a Neural Network
0	Initialize and determine best training parameters
1	Change lower case characters to upper case
2	Eliminate non-alphabetic characters
3	Eliminate non-relevant words
4	Eliminate non-relevant word endings
5	Determine n-grams
6	Select relevant n-grams
7	Determine vectors representing relevant n-grams
6	Teach vectors to Kohonen feature map
	Step B: Pass Free Text along Neural Filter
0	Determine start and end of texts (line, paragraph, separator, etc.)
1	Change lower case characters to upper case
2	Eliminate non-alphabetic characters
3	Eliminate non-relevant words
4	Eliminate non-relevant word endings
5	Determine n-grams
6	Select relevant n-grams
5	Determine vector representing relevant n-grams
6	For every text: for every vector: input vector to neural network and determine error
7	Select text if cumulative error < threshold

3.4 Algorithm 2.1: The Neural Interest Map Based on Keyword Clustering

The algorithms just mentioned use the Kohonen feature maps in a way they were not intended to be used. The maps are only used to remember groups of n-grams or keywords which occur with a certain conditional probability (these n-grams are not necessarily the most frequent ones!). There are two types of conditionalities involved in the model. First there is the conditional dependencies derived from the shifting window. Second (and more important), there is the conditionality resulting from the recurrent effects within the feature map (see the section "Results for the Neural Filter Algorithm Based on N-Grams" on page 164 for a more detailed discussion on this subject). The resulting topology of the feature map is ignored completely (See the section Results for a more thorough deliberation on this topic).

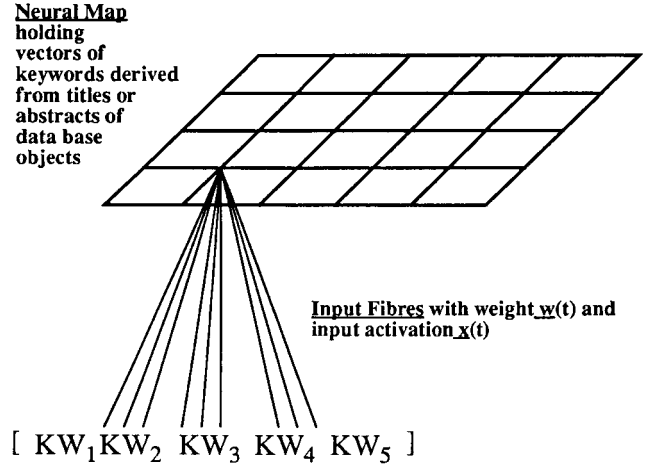


FIGURE 6. The neural interest map based on keyword clustering

Another, more common use of feature maps is clustering of keywords that represent interests. Assume a full-text data base and a limited vocabulary in a specific domain (about 1,000 words). Then, each text in the data base can be represented by a vector holding a dimension for the frequency of every keyword (see figure 6, "The neural interest map based on keyword clustering"). By teaching the keyword vector for every text to the Kohonen feature map, a topological representation of various interests will occur (see Table 4, "Algorithm 2.1: The interest map based on keyword clustering"). Such a map might be seen as a neural interest map, where related papers are clustered in adjacent neighborhoods. The main difference between this method and work done by [Lin et al., 1991] is that our model uses the full text (or that of an abstract) to cluster the papers,

where Lin only uses 25 keywords occurring in paper titles. The number of keywords used here is much larger (500).

The map formed might be seen as a semantic map of the data base texts. Since [Doyle, 1961] there has been research towards the automatic formation of such maps. Doyle expressed his desire to use the computer not only as a tool in searching, but as a method of discovering semantical relations. This approach is quite similar to the neural network formalism of Kohonen. [Ritter et al., 1989b], [Ritter et al., 1990] and [Ritter, 1991] show possible applications of such self-organizing sematopical maps in the derivation of semantic relations between words. Moreover, there is literature on the functional specifications of a user friendly interface for document relations [Crough, 1986]. The specifications pointed out here strongly resemble the characteristics of the Kohonen feature maps.Although the relation between the cognitive and semantic maps as meant in the literature and the Kohonen formalism is not that direct, the Kohonen feature maps do share some properties of cognitive maps. Kohonen maps express relations between objects in euclidean distances, and they are able to reduce complex relations in an n-dimensional feature space into a lower two- (or three-) dimensional space with conservation of spatial and topological relations. More on research toward the cognitive map can be found in [Lakoff, 1988], [Regier, 1988], [Chrisley, 1990], and [Palakal et al., 1991]

TABLE 4. Algorithm 2.1: The interest map based on keyword clustering

	Step A: Teach interest map to a Neural Network
	For all data base objects do:
0	Change lower case characters to upper case
1	Eliminate non-alphabetic characters
2	Eliminate non-relevant words (uniformly frequently occurring)
3	Eliminate non-relevant word endings
4	Determine keywords (within text frequently occurring)
	For all keyword vectors do
5	Determine best training parameters
6	Teach vector to Kohonen feature map
	Step B: Map Query with Neural Interest Map
0	Determine start and end of texts (line, paragraph, separator, etc.)
1	Change lower case characters to upper case
2	Eliminate non-alphabetic characters
3	Eliminate non-relevant words
4	Eliminate non-relevant word endings
5	Determine selected keywords
6	Input selected keyword vector to the neural network and determine error of position on map that is maximally activated
7	Select data base objects with distance D from this position if retrieval error < threshold

3.5 Algorithm 2.2: The Neural Interest Map Based on Character N-Gram Clustering

Suppose one has calculated an n-gram vector holding the k most frequent n-grams for a certain (static) text stored in the data base. Then, a way of comparing a query to the data base is by comparing the n-gram vector with all the n-gram vectors in the data base. If the data base contains many texts, this might be quite a job. Therefore, statistical methods use clustering algorithms and compare the n-gram vector to a cluster of data base elements. By teaching the n-gram vectors of the data base to a Kohonen feature map, a topological (clustered) map of interests develops automatically. To query the data base, a free-text query is processed like the training text. The resulting vector is positioned on the map. By investigating the activity on the map, the area representing this vector can be found efficiently. All texts represented by neurons in the neighboring region can then be considered of interest. A threshold function fine tunes the system.

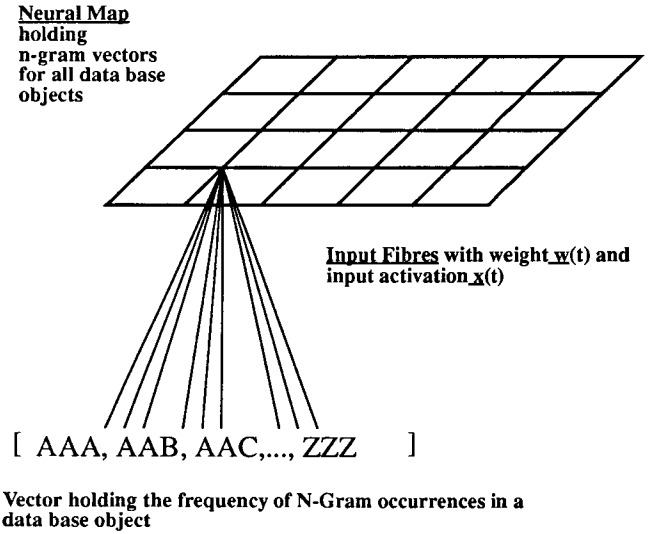


FIGURE 7. The neural interest map based on n-gram clustering

In the case of a 3rd order Markov chain over 27 characters, these vectors have dimension $27^3 = 19{,}683$, which is definitely too much for any practical solution. But, one can transform this vector to a much smaller base, without losing too much of its characteristics. The legitimacy of this dimension reduction is based on the heuristic that most elements in the trigram vector $\underline{y}(t)$ are about zero. Suppose the new basis consists of vectors ζ_i. Then, a number of vectors ζ_i are generated randomly, one for each dimension of $\underline{y}(t)$. The reduced

vector $\underline{y}'(t)$, is determined by computing the sum of the products of the separate dimensions of $\underline{y}(t)$ with the corresponding components of in ζ_i.

$$\underline{y}'(t) = \sum_{i=1}^{N} \underline{y}_i(t) \cdot \zeta_i \qquad \text{(EQ 1)}$$

where

N = Dimension vector $\underline{y}_i(t)$

$\underline{y}_i(t)$ = Original vector with dimension N

ζ_i =Vector i from a set of N vectors of lower dimension n which altogether form the new basis for $\underline{y}'(t)$

$\underline{y}'(t)$ = Transformed vector of lower dimension n

By doing so, the number of fibers can be reduced enormously. So, even large maps derive complex representations within reasonable time limits. See [Ritter et al., 1989a] for a proof of the legitimacy of this operation.

After dimension reduction, a 500 to 1000 dimensional vector remains, which represents all possible keyword relations without any dictionary and prefix- or suffix stripping. Although this method still uses complicated algorithms to determine the initial n-gram vectors and to reduce them in dimension, the neural network smoothly solves the entire generalization and association process (see Table 5, "Algorithm 2.2: Interest map based on trigram clustering").

TABLE 5. Algorithm 2.2: Interest map based on trigram clustering

	Step A: Teach Interest Map to a Neural Network
	For all data base objects do:
0	Change lower case characters to upper case
1	Eliminate non-alphabetic characters
2	Eliminate non-relevant words
3	Eliminate non-relevant word endings
4	Determine n-gram tables, select the *n* most frequent tables
	For all n-gram vectors do:
6	Reduce dimension
7	Teach n-gram vector to Kohonen feature map
	Step B: Match Query with Neural Interest Map
0	Determine start and end of texts (line, paragraph, separator, etc.)
1	Change lower case characters to upper case
2	Eliminate non-alphabetic characters

3	Eliminate non-relevant words
4	Eliminate non-relevant word endings
5	Determine n-gram vector, select the n most frequent elements
6	Reduce Dimension
7	Input n-gram vector to neural network and determine error of position activated
8	Select data base objects with distance *D* from this position if the error < threshold

3.6 Algorithm 2.3: The Neural Interest Map Based on Dynamically Chosen N-Grams

Now, what if the keywords are too limited (due to a dynamic and unknown vocabulary) and the trigram vectors end up having far less zero elements than expected (see figure 8, "The neural interest map based on dynamically chosen n-gram clustering"). Then, one can derive the most frequent (large) n-grams and teach the Kohonen feature map a vector representing a specific n-gram in every dimension. After training, the feature map represents an interest map of the full-text data base. Objects related to each other in n-gram use are within nearby clusters (see Table 5, "Algorithm 2.2: Interest map based on trigram clustering").

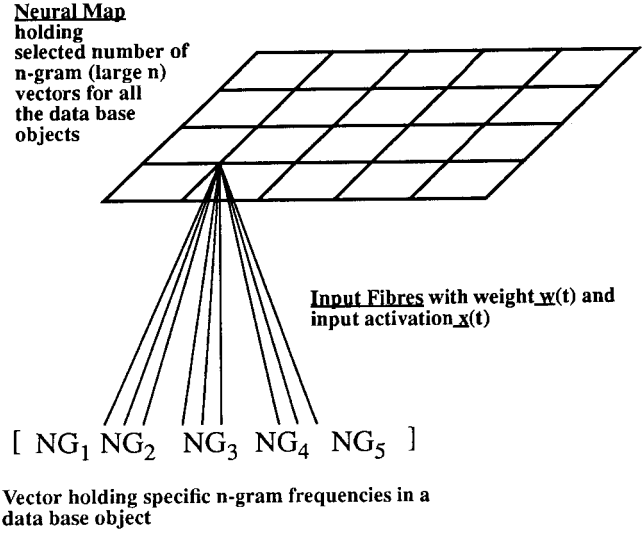


FIGURE 8. The neural interest map based on dynamically chosen n-gram clustering

TABLE 6. Algorithm 2.3: Interest map based on (large preselected) n-gram clustering

	Step A: Teach Interest Map to a Neural Network
	For data base objects do:
0	Change lower case characters to upper case
1	Eliminate non-alphabetic characters
2	Eliminate non-relevant words
3	Eliminate non-relevant word endings
4	Determine n-gram tables, select the n most frequent tables
	For all n-gram vectors do:
5	Teach n-gram vector to Kohonen feature map
	Step B: Match Query with Neural Interest Map
0	Determine start and end of texts (line, paragraph, separator, etc.)
1	Change lower case characters to upper case
2	Eliminate non-alphabetic characters
3	Eliminate non-relevant words
4	Eliminate non-relevant word endings
5	Determine n-gram vector, select the n most frequent elements
6	Input n-gram vector to neural network and determine error of position activated
7	Select data base objects with distance D from this position if the error < threshold

4.0 Training and Retrieval Rules

The training rule used in the previous model is the Kohonen rule. It does not matter whether one uses characters, words or large n-grams as input elements, a coding procedure prepares all symbolic data for input to the feature map by translating it to vectors. This coding process is performed with the aid of a lookup table. All elements of the training set are assigned randomly to specific codes in this lookup table. The codes themselves are spread homogeneously through the feature space, to speed up the training process. Convergence parameters as proposed by [Ritter et al., 1989a] fine tune the Kohonen rule. For the sake of completeness, the Kohonen training rule is repeated here[1].

4.1 The Kohonen Training Rule

First, determine the neuron v, which has the Best Match between the input values and its weight values:

$$v: \qquad \forall k \left(\left\| \underline{w}_v(t) - \underline{x}(t) \right\| \leq \left\| \underline{w}_k(t) - \underline{x}(t) \right\| \right) \tag{EQ 2}$$

for k an neuron on the feature map

Next, update all weights in the map according to the Kohonen training rule:

$$\underline{w}_k(t+1) = \underline{w}_k(t) + \varepsilon(t) \cdot \Phi_{kv} \cdot (\underline{x}(t) - \underline{w}_k(t)) \tag{EQ 3}$$

where

$$\Phi_{rs} = e^{-\| k - v \| \cdot 2\sigma^2(t)} \tag{EQ 4}$$

$$\varepsilon(t) = \varepsilon_{max} \cdot \left(\frac{\varepsilon_{min}}{\varepsilon_{max}} \right)^{\frac{t}{t_{max}}} \tag{EQ 5}$$

$$\sigma(t) = \sigma_{max} \cdot \left(\frac{\sigma_{min}}{\sigma_{max}} \right)^{\frac{t}{t_{max}}} \tag{EQ 6}$$

1. Kohonen Feature Maps are discussed in depth in Chapter 1: "Neural Computation" and in Chapter 3: "Recurrent Kohonen Feature Maps in Natural Language Processing".

where

$\omega \in [0,1]$ Memory Rate,

$\varepsilon_{max} \in [0,1]$ Initial Learning Rate,

$\varepsilon_{min} = [0,1]$ Final Learning Rate,

$\sigma_{max} \in [0,1]$ Initial Region Size

$\sigma_{min} \in [0,1]$ Final Region Size, and

$\| k - v \|$ Physical distance between neuron k and neuron v

4.2 Retrieval Rules

Once the feature map training is completed, one must match the test data with the representation formed on the neural map. In the case of the neural filter, one counts the cumulative (normalized) error or the cumulative (normalized) number of perfect hits (or some variant of these two functions).

In general, one can separate two types of selection rules: *positive* and *negative* ones. The negative approach mainly filters the noise. A more positive approach is to choose possible candidates for selection. Negative selections are mostly normalized, while positive ones are not. If one paragraph in a paper is related to a specific interest, the positive filter selects it directly, where the negative one ignores the one paragraph due to normalization of the retrieval value (one paragraph fires high, all the others low, so the average firing level is still low). Positive selection mostly results in too many candidates where negative selection results in too few candidates. A proper combination of both approaches results in the best retrieval results.

Possible positive search methods are plain keyword matches and the (non-normalized) number of perfect hits on the neural map (in the case of n-gram on characters as well as n-gram on words). A negative filter is the added and normalized error of all text elements with respect to a statistical table or a neural map.

4.2.1 Selection Rule 1 for Neural Filter (Negative)

The Negative selection rule eliminates the most irrelevant documents with respect to the filter. A document is selected if and only if:

$$\frac{\sum_{i=1}^{N} \| \underline{w}_s(t) - \underline{x}_i(t) \|}{N} < \tau \qquad \text{(EQ 7)}$$

where s has the property:

$$\forall r \left(\| w_s(t) - x_i(t) \| \leq \| w_r(t) - x_i(t) \| \right) \qquad \text{(EQ 8)}$$

for:

r element of the feature map

$\tau \in [0,1]$, threshold value before an n-gram is taken into consideration

$\underline{x}_i(t)$ is a vector holding one n-gram of keywords, characters or dynamically chosen n-grams (direct coding through look up tables)

4.2.2 Selection Rule 2 for Neural Filter (Positive)

The positive filter is used to retrieve those documents which are *closely* related to the neural filter. A document is selected if and only if:

$$Count \left(\frac{\sum_{i=1}^{N} \| \underline{w}_s(t) - \underline{x}_i(t) \|}{N} < \tau \right) > \phi \qquad \text{(EQ 9)}$$

where s has the property:

$$\forall r \left(\| w_s(t) - x_i(t) \| \leq \| w_r(t) - x_i(t) \| \right) \qquad \text{(EQ 10)}$$

for:

r element of the feature map

Count(x) is a function that counts all occurrences which fulfill condition x

$\tau \in [0,1]$, threshold value before an n-gram is counted (small value)

$\phi \in [0,1]$, threshold value before a document is selected on basis of the counted n-grams

$\underline{x}_i(t)$ is a vector holding one n-gram of keywords, characters or dynamically chosen n-grams (direct coding through lookup tables).

One may combine the positive and negative retrieval rule in order to obtain an optimal mechanism.

4.2.3 Selection Rule for Keyword Neural Interest Map

With the interest map, one determines the neuron representing the interest vector best and returns the paper represented by this neuron and all the other papers within the same cluster (determined by euclidean distance or by knowledge of the cluster boundaries on the feature map).

The selection rule for the "keyword based neural interest map" yields: all objects represented by neighboring neurons of neuron s, where s has the property:

$$\forall r \left(\| w_s(t) - x(t) \| \leq \| w_r(t) - x(t) \| \right) \qquad \text{(EQ 11)}$$

and

$$\| w_r(t) - x(t) \| < \tau \qquad \text{(EQ 12)}$$

for:

r element of the feature map

$\tau \in [0,1]$, threshold value, only query results higher than this threshold are taken in consideration

$\underline{x}(t)$ is a vector holding keyword frequencies. One dimension for every keyword known by the system.

4.2.4 Selection Rule for N-Gram Neural Interest Map

The selection rule for the "n-gram based neural interest map" yields: all objects represented by neighboring neurons of neuron s, where s has the property:

$$\forall r \left(\| w_s(t) - x(t) \| \leq \| w_r(t) - x(t) \| \right) \qquad \text{(EQ 13)}$$

and

$$\| w_r(t) - x(t) \| < \tau \qquad \text{(EQ 14)}$$

for:

r element of M

$\tau \in [0,1]$, threshold value, only query results higher than this threshold are taken in consideration

$\underline{x}(t)$ is a vector holding n-gram frequencies. One dimension for every n-gram known by the system.

4.2.5 Selection Rule for Selected (Large) N-Gram Neural Interest Map

The selection rule for the "selected (large) n-gram based neural interest map" yields: all objects represented by neighboring neurons of neuron s, where s has the property:

$$\forall r \left(\| w_s(t) - x(t) \| \leq \| w_r(t) - x(t) \| \right) \quad \text{(EQ 15)}$$

and

$$\| w_r(t) - x(t) \| < \tau \quad \text{(EQ 16)}$$

for:

r element of the feature map

$\tau \in [0,1]$, threshold value, only query results higher than this threshold are taken in consideration

$\underline{x}(t)$ is a vector holding frequencies of preselected n-grams. One dimension for every n-gram known by the system.

5.0 Simulations and Results Neural Filter

The simulations are implemented on a high end PC (33 Mhz 386) and on a Sun Sparc Station IPC (The Neural Filter runs on the PC as well as on the Sun IPC. The Neural Interest Map runs only on the Sun IPC due to its huge memory requirements). The programs are written in C. The PC was connected to a CD-Rom player holding several free-text data bases such as: The Complete Works of Shakespeare, The Complete Sherlock Holmes, and The Complete Translated 1987 Pravda articles. The most intriguing one was the Pravda data base, which shall be used in the examples of this chapter.

The simulation parameters are determined automatically according to the best parameter heuristics in [Ritter et al., 1989a] and [Scholtes, 1991m]. Before training, the training text was analyzed to provide the optimal values for the internal coding, the learning rate, the region size, and the approximate number of necessary training cycles to reach the self-organizing state.

In the case of algorithm 1: the Neural Filter, the training set holds a small selection on the 1987 nuclear weapon restriction talks between the USA and the USSR. The test set was the entire Pravda CD-Rom (200 Mbyte), being passed along the neural filter.

> "Our era, a fast-paced era of nuclear weapons, an era of growing economic and political interdependence, precludes the possibility of security for one nation at the expense of others. I repeat: we can only survive or perish together. Security today can only be viewed as mutual, or to be more precise, universal. So whether we like each other or not, we need to learn how to coexist and live in peace on this small and very fragile planet. Question: Do you support the continuation in 1987 of the Geneva talks between Soviet and American representatives for the purpose of achieving progress on the issue of limiting and reducing arms? Answer: Yes, we do. We support talks that would overcome the state of fruitlessness and inertness and acquire true dynamism, in a word, talks that would become genuine talks on reducing nuclear arms and preventing an arms race in space. We tried to achieve that in Reykjavik and will try to achieve it even more energetically in 1987. I am sure that such a radical turnaround in the talks woul respond to the vital interests of the American people as well. At the same time, the position of the US administration on this issue is a cause of great disappointment for us. After Reykjavik the American delegation in Geneva has become even less cooperative. Despite the fact that the USSR has not been conducting nuclear detonations for 18 months, the USA has continued tests and refused to discuss a total ban on them, though it committed itself to conduct negotiations on that issue in the two treaties of 1963 and 1974. In November that was aggravated by the provocative action the White House took when it broke the important strategic arms limitaion agreement (SALT II). It does not help to conduct successful negotiations on new agreements when the old ones are being deliberately and blatantly broken. This is a serious problem that deserves very close attention. I will state once again that we support agreements on the most radical reductions of arms, both nuclear and conventional. Now it is up to Washington."

FIGURE 9. Training set (or 'query')

Before training, the text was filtered to avoid wasting computation time (or neural memory) to non relevant features. First all non-alphabetic characters (digits, read markers, etc.) were eliminated. Next all lower case characters were transformed to upper case. About 250 common English words were eliminated from the text. To obtain an even stronger distinguishing behavior, all common endings (resulting in trigrams such as *-ary, -able*, and *-ent*) were eliminated too. A number of such common words and word endings can be found in the tables below.

TABLE 7. Part of common words table

a	might	no
few	always	both
little	going	here
about	much	not
first	good	by
many	near	I
after	at	off
get	got	end
me	never	is
all	been	often
go	he	even
and	new	it
just	before	once
she	her	every

TABLE 8. Part of common endings table

Ending
-ably
-ibly
-ily
-ss
-ous
-ies
-s
-ied
-ed
-ing
-...

5.1 A First Statistical Analysis of the Training Set

A first analysis made was the determination of the 27 most frequent trigrams in the query by brute force counting. After processing the text according to the method presented in Table 1, "Algorithm 1.1: Neural filter with n-grams", the table below holds some values found in alphabetical order. One can spot the trigrams from words as *Weapons* and *Nuclear* right away. This table is used for some global comparison between the statistical and neural methods.

Trigram	Frequency	Weighted frequency	Trigram	Frequency	Weighted frequency
AGR	36	0.005380	ITY	40	0.005977
AIN	36	0.005380	LEA	56	0.008368
ARM	40	0.005977	MEN	60	0.008966
CLE	52	0.007770	NUC	48	0.007173
CON	48	0.007173	PEA	36	0.005380
DUC	40	0.005977	PER	36	0.005380
EAC	40	0.005977	POS	36	0.005380
EAR	60	0.008966	PRO	48	0.007173
EAT	44	0.006575	REA	60	0.008966
ENT	84	0.012552	STR	48	0.007173
ERI	40	0.005977	TER	40	0.005977
EST	44	0.006575	TRE	36	0.005380
GRE	48	0.007173	UCL	48	0.007173
INT	44	0.006575			

FIGURE 10. Most frequent trigrams in training text

5.2 Result for the Neural Filter Algorithms

In this section, the results obtained in the neural filter simulation shall be discussed. Different cases are considered: with and without spaces, higher and lower order n-grams, and positive and negative selection functions.

5.2.1 Results for the Neural Filter Algorithm Based on N-Grams

The following simulation filtered the query and taught the trigrams (3-gram) up to 4 times to a 15 by 15 Kohonen feature map. Because the text contained many more trigrams than the 225 neurons could hold, only the most dominant (according to frequency and distance in internal coding) ones were stored and each trigram was only held by one neuron. The presented trigram map was obtained by determining the code for each neuron after the

training process had converged to a stable state. The meaning of the neuronal fibres can be looked up rapidly in the internal coding tables. The trigrams shown below are the best ones. It could well be that a specific neuron represents other trigrams as well within certain limits. By comparing the most frequent trigrams obtained with the statistical method, one can observe that the trigrams represented by the neurons in the map are indeed among the most frequent ones. The presented trigram map evolves by determining the code for each neuron after the training process converged to a stable state. The trigrams for NUCLEAR are underlined as an example. In this one and all the following maps, XXX means no proper symbol could be found (see figure 11, "Trigram feature map of query in neural filter").

ERN	MOM	LLY	PLA	HUM	ANI	UGG	TRY	LIM	TRA	STA	MPS	GTO	ENO	PRO
DOU	IMA	LAN	URT	REL	IMP	UPP	YME	LEA	LUD	KJA	TUA	YTH	ION	GIC
CUM	ANT	ITS	RAT	LIZ	SEE	TOU	ODW	MBE	EME	UTU	SSF	STR	DOF	POL
STS	MEN	ENT	EAR	REA	ICY	THE	ORY	DUC	ATE	ATY	APO	SUP	SOV	SEL
END	DIS	PHE	OSS	TAS	OAC	XIB	ELA	GRA	HIG	OLI	ARM	WID	ARY	ILI
GRE	ONT	UMS	LIT	UCT	UCL	REQ	TLY	BOD	ICA	QUI	NOB	ORI	GIL	FIN
ALL	EQU	BIL	ARC	EEP	UNC	OVO	CON	CLE	ISC	DET	OGR	OFF	RIT	ECI
AST	ALT	HIE	VOC	MPL	LAB	MAT	GLI	CLI	INT	RCH	RSE	OPO	RIZ	RAN
EST	OSP	UNR	PAC	IRE	OMP	TIP	OUS	NSI	ITI	OYM	INI	INC	IZO	UMA
RST	SSU	SHI	PEA	YST	NTE	OUR	NTA	BOR	NTO	ITY	CHI	EGI	OOD	OBO
NIT	USY	SET	EEM	PSE	ANG	ISA	EYK	AVI	NCL	CLU	BEC	HOR	HER	DAY
TIC	SPH	TUR	TIO	JAV	OPH	OVI	APP	ATO	SUC	RUG	NUC	VIK	DEE	ORT
TMO	TES	TSE	GER	SAR	MAI	MAL	ART	ANC	ILL	ZON	YKJ	xxx	xxx	AGR
ROP	FIC	SPL	VOR	EAR	MAN	REY	TON	LAT	ABO	NDO	DUR	NYT	IDE	OME
TEG	UIR	TER	UCI	GGR	ABM	AGA	EAC	TAN	ACE	ALI	MUT	GGL	GOT	BLY

FIGURE 11. Trigram feature map of query in neural filter

There are some aspects that are still not completely clear in this phase of the research. If one compares the most frequent trigrams (according to the statistics) with the trigrams in the feature map, then some of them are missing: *ain, eat, eri, per, pos, tre.* On the other hand, some trigrams occur that cannot be found in the training text at all, such as: *ggl, xib.* This is probably due to the neighborhood effects, which do indeed absorb less frequent trigrams, but they also disturb the internal coding of existing trigrams and cause the occurrence of non-existing trigrams.

These two phenomena are more clear in the following examples. Suppose one trains a single occurrence of the trigrams *aaa, aab, ..., aaz* and 10 trigrams *zzz*, and the Kohonen map consists of 1 neuron. Then, only *aaz* will be represented and not *zzz*. Although *zzz* is the most frequent one. But, the *aa...* combination occurs more frequently [1].

1. I owe the discovery of these effects and the examples to Brian Bartell.

This effect can be defended as being positive, because the feature map does not only remember the most frequent trigrams, but it also takes the context in which the n-grams occur into account. In other words, Kohonen feature maps as they are used here, implement a technique to derive conditional probabilities with respect to the other n-grams trained to the feature map [1].

The other effect is more serious. Say one trains many *aaa*'s and *aac*'s, then the model probably ends up in *aab* if it is constructed of one neuron, although *aab* never occurred! This effect explains the occurrence of non-existing trigrams in the feature map. However, the effect occurs only if the recognition threshold is too large. By sharpening the threshold, this effect might appear in theory, but practical emergence will be limited to rare cases. In fact, the trigrams *ggl* and *xib* in the above feature map had pretty large errors (>5%).

Normally, it is not worth the trouble to derive 4-, 5-, 6-, and 7-grams. One often encounters the idea that trigrams give more than enough separation. This is true if an increase of the window size corresponds to an exponential increase in the space or the time needed to derive such dependencies. However, here one sees a linear increase of complexity as the window size grows (as shall be shown further on). This result makes it very easy and worth the trouble to derive higher order n-grams and determine their influence on the retrieval process.

An advantage of neural networks over statistics follows undoubtedly from the following simulation. By increasing the window size from 3 to 4, the neural network learns 4-grams within almost the same amount of memory and at the same speed as the trigram implementation. However, the statistical method needs either a factor 27 more memory, or decreases dramatically in speed. One can recognize frequent words from the training text even better than in the trigram map (see figure 12, "4-Gram feature map of query in neural filter").

1. One might even state that the values derived in the feature maps relate to Shannon probabilities. More on this subject can be found in Chapter 5: "Neural Data-Oriented Parsing".

```
EREO RUGG LIMP LITI CAUS PLOY DISA EOTY GREA TEND ALIZ WISH EATY LOYM
VIET GLIM ANYT NUCL GOOD XTEN TRIC IGUR COOP SINC WILL ACHI DSHI SECU
ONTO APPR CURI TOUR OGRA MPRO TANC ATEL FRIE SAPP ABIL AINT EACE EYKJ
RTUN PROP NCER WEST IZON ENDS LICY ARCH UCIN SPAC ABOR VERN NALL OPOS
AVIK ALLY NCLU OUSY STAN ISAR HUMA UNAT STRI EDUC UCLE REOT AREA SELF
TSEL KJAV CLEA BORA OSPH MSTA RIEN NDSH NMEN CATA UTUA RELA LIMA REYK
ICAN SHIP EACT RONG MPLE ATEG TRUG ASTR TIPU POLI FIGU OLIC YKJA ORIZ
MENT LEAR AGAI SOVI MOSP OOPE USSR JAVI OTYP CLUD IMPR DOFF IEND SPHE
PSET ITHD ORTU PAST UBLY FICI EXTE TUNA POSS ERNM PEAC UNFO ESEA INCE
OSIT CUMS MUTU STIP SCUS BODY EGIC UGGL REAC IMPS STRO REAT NTOU AGRE
ISSU REST GOVE ERST PHER OPER GTON OLIT TLES SUPP xxxx xxxx xxxx xxxx
```

FIGURE 12. 4-Gram feature map of query in neural filter

The same holds for the 5-gram simulation: speed and memory requirements were about the same as for the trigram simulation. The possibility to represent 5-grams almost eliminates the need to develop a large dictionary of known words. 5-grams can be used in any language without a-priori knowledge.

```
STABI TRICT PERIT ALLIS REOTY DERST REATY TSELF SECUR MAINT VERYT ALISM
TEGIC POLIC OLITI PARTY ANTLY AGREE POWER JAVIK ECURI ORIZO NFORT CIRCU
NSIDE OMPLE STRUC CURIT ACEFU ETELY LITAR MMEDI REDUC EACEF NDOFF NDSHI
ATEST SCUSS GUARD UCLEA EDUCT WERFU EREOT EMAIN FIRML RDINA UALIT WEAKN
MPROV RELAT REMAI COURT BORAT PURSU SUPPO SSARY RUGGL PLETE PEACE EFEND
ESSAR RADIC DEVEL OWERF XTRAO SOVIE QUALI CESSA ORTUN EARCH ISAPP TUNAT
UMSTA ANGER EYKJA URITY DESTR VELOP THDRA CLEAR FRAGI HUMAN OPMEN SINCE
TRUCT ERFUL TASTR NEGOT IENDS EOTYP ESTRU CIALI LABOR DEFEN ILITA RTUNA
OBODY TIPUL IALIS ATEGI LITIC XIBIL COMPL OVIET EGOTI MALIZ ECESS LIMIN
ARTIC PMENT QUIPP GOODW KJAVI MPLET YKJAV AKNES IRCUM INCLU BLATA UMANI
PRINC WEAPO STRIC VERNM UCING RMITT XTEND LIMPS TMOSP ONTOU PURPO HORIZ
SOCIA APPRO CLIMA ERICA EAPON OCIAL EVERY IRMLY DISPL ERYTH NECES EQUIP
EVELO EQUAL STAND PROPO REYKJ OSPHE xxxxx xxxxx xxxxx xxxxx xxxxx xxxxx
```

FIGURE 13. 5-Gram feature map of query in neural filter

To convince the reader even more, the next two pictures hold n-gram feature maps for 6- and 7-grams. All calculated within the same amount of memory and at the same speed.

```
xxxxxxx RIENDS  ROSPER  SECURI  xxxxxxx COURSE  xxxxxxx SOVIET  ECURIT
xxxxxxx xxxxxxx xxxxxx NEGOTI  xxxxxxx TREATY  ANDOFF  ONTOUR  OLITIC
OCIALI  LIMATE  xxxxxxx xxxxxxx WERFUL  EQUIPP  PARTIE  APPROA  SOCIAL
xxxxxxx IALISM  CIRCUM  SEARCH  NDSHIP  ROGRAM  INCIPL  REDUCT  MSTANC
TURNAR  xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx ORDINA  REYKJA
STRONG  INCLUD  ITSELF  POWERF  xxxxxxx IMMEDI  xxxxxxx ACEFUL  xxxxxxx
ATIONS  xxxxxxx TRATEG  SPERIT  xxxxxxx RMITTE  PEACEF  KJAVIK  DANGER
```

```
STEREO  xxxxxxx POLITI  xxxxxxx ODWILL  xxxxxxx xxxxxxx TROPHE  OWERFU
xxxxxxx DISPLA  xxxxxxx ACHIEV  ASTROP  xxxxxxx DEVELO  xxxxxxx ELOPME
xxxxxxx QUALIT  PROPOS  CURITY  IRCUMS  xxxxxxx NSIDER  HIGHER  xxxxxxx
MERICA  xxxxxxx xxxxxxx ISAPPO  YKJAVI  HORIZO  CUMSTA  xxxxxxx RGETIC
xxxxxxx TANTLY  REMAIN  NOBODY  DEFEND  EQUALI  VIOUSY  IENDSH  REOTYP
FIRMLY  SUPPOR  ECESSA  CONNEC  DEPLOY  ORTUNA  ENOUNC  PPROAC  HINGTO
UCLEAR  EVERYT  PPOINT  xxxxxxx ATMOSP  xxxxxxx xxxxxxx PROGRA  xxxxxxx
EYKJAV  MPLETE  HUMANI  PROSPE  VERYTH  xxxxxx TUNATE  xxxxxxx SHINGT
UNATEL  NUCLEA  RTUNAT  CLIMAT  FRIEND  PRINCI  OSPHER  ITHDRA  GLIMPS
```

FIGURE 14. 6-Gram feature map of query in neural filter

```
ENDSHIP xxxxxxx xxxxxxx xxxxxxx xxxxxxx PRINCIP xxxxxxx ERGETIC POLITIC
GREATES EQUALIT OMPLETE AINTAIN RESTRIC ENEFICI xxxxxxx xxxxxxx PROSPER
HINGTON xxxxxxx MAINTAI PARTIES PERMITT SUCCESS xxxxxxx xxxxxxx ADMINIS
RDINARY CONNECT POWERFU YKJAVIK xxxxxxx xxxxxxx xxxxxxx AORDINA NTEREST
xxxxxxx xxxxxxx BVIOUSY REDUCIN PROGRAM xxxxxxx EVERYTH NTIBALL STRATEG
UTUALLY xxxxxxx NECESSA RCUMSTA xxxxxxx xxxxxxx xxxxxxx QUALITY xxxxxxx
DYNAMIS NRESTRI NDEPLOY xxxxxxx ISAPPOI ITHDRAW xxxxxxx FRIENDS xxxxxxx
xxxxxxx EREOTYP REYKJAV ECESSAR TUNATEL TANDOFF TASTROP xxxxxxx xxxxxxx
IRCUMST xxxxxxx COMPROM xxxxxxx YNAMISM PEACEFU LATIONS xxxxxxx xxxxxxx
ECURITY ESEARCH xxxxxxx FLEXIBI TRUMENT CONTOUR xxxxxxx AMERICA xxxxxxx
TMOSPHE IENDSHI ASTROPH xxxxxxx DISPLAY DISAPPO NUCLEAR ROSPERI EACEFUL
xxxxxxx xxxxxxx APPOINT COMPLET CUMSTAN ATANTLY ATMOSPH xxxxxxx MERICAN
NTIRELY EEMENTS xxxxxxx RIENDSH LLISTIC ANYTHIN OWERFUL xxxxxxx ATASTRO
UMSTANC APPROAC PPOINTM SOCIALI CLIMATE xxxxxxx UCCESSF TLESSNE SECURIT
OSPERIT OVEMBER xxxxxxx EYKJAVI xxxxxxx AGREEME xxxxxxx xxxxxxx xxxxxxx
xxxxxxx xxxxxxx STEREOT SUPPORT STROPHE ORDINAR xxxxxxx SHINGTO PPROACH
```

FIGURE 15. 7-Gram feature map of query in neural filter

In all the above simulations, the best size for n seemed to be the average word length of the language ("best" means the most efficient trade off between computational efforts and retrieval or representation quality).

5.2.2 Results for the Neural Filter Algorithm Based on N-Grams with Spaces

The n-grams as shown above are all without the incorporation of spaces. In the next simulations the spaces were used too, to gain a better insight in the contextual relations between words. Without spaces one actually only determines keyword parts. By incorporating the spaces, relations between words are taken into account as well. Without these relations, n-grams are nothing more than a keyword search method without a dictionary of predetermined keywords. The spaces are represented by [.

```
xxx [EV [EQ WAS CAT BER RYT WID SYS [NE [MA TIC WES T[W K[C L[R INK INC
MOS OOD DAN [AT [AB LET xxx SEC FAC UES FIG RIC HER TER TES xxx POW PPR
GOT END INT OIC ONS NEG OAC NES OUR xxx EAT NAT MED IEV GUR VER RAW FAV
xxx xxx AND UNR ENT NIT OOP N[G N[S LIS NEC ACT UCT IET URT FUS UAR EAR
EQU EGI IMA EFU A[W ANG ANS FIR I[T IPP Y[A H[W RST EXT ITS NST UCC ARD
ERF EAS EVE M[A ELA ACE CIA QUA RMA VIE NGT Y[G F[D LOV GOV ARC NDS [ST
ECT ESS ERS EAC xxx I[E MLY xxx CEF RMI ANI ENE L[T G[S T[R D[F xxx ORC
OSS NTO [SH IRC YTH xxx ABO AKN AIN EEM REY RML UNF UNA GET [IT C[S GGR
DEF CUR [CL [EL [SO [FO GGL EGO ATO ISP EYK MEN MAI FUL IMM EMA [EX LIC
DIS DUC CES CIP TRO GTO OSP [DO xxx UPP EXP EAP UIP MIN YKJ ION MOM IOU
LUD [US RUC HAL TEL TMO BAL PRO OBO [CO V[P ELO ALK ACH IZO Y[N YON N[N
NOU MPS PUL ERO REL REO TYP RYO GEN LAY ORM [AM xxx CHO S[H UCL UGG ILL
[PO OPO ROP OOR NFO IAL EAK FEN RAN HIN LIM LIZ [IN [UN [ON SHO C[E AGG
RCO DOO OMP L[L LOP D[P N[P IGH xxx xxx CUM RIN PEN HUM [MI [MO TON T[N
DWI THI C[N CON RNM H[N SPH SOV xxx RTH SSR SSU ONN GAI DMI GRE xxx SOC
T[I XTE TSE RTU SHI V[I R[N R[U DOF ROW xxx DSH HDR [GU HOI [FI [RE FRI
FRU SWE SAR TRA TRI TEN CCE ALI R[I T[U TOU BOD [TR [DE [HA [LI L[U [SU
[WE ORA [GA C[C SSF TRU RFU xxx SSA SPE XPE T[A D[G TOT NCE OCI NTE KJA
PEA LAB [QU [WA A[B ALT xxx ARY AVI REM QUI R[E C[A T[F THD T[D OTI NGE
OKE NNE OYM PER NER GUA Y[R xxx ARM ISM URI ERN ISA RSE HIE S[G R[T R[S
LLY PLE [LA PME OVI OFF OGR [FR ELS Y[D AGR ITA ATM USY ICY GHE GLI xxx
R[R U[F OPM OPE [PE M[F PRI PSE OWE [PR FOR BOR ECE ATI USA ITY ETE FLE
```

FIGURE 16. 3-Gram with spaces feature map of query in neural filter

As in the earlier simulations, the window size can be increased easily to 7-grams (or higher).

5.2.3 Results of Neural Filter Algorithm based on a Markov Chain over Keywords and N-Grams. [1]

Now, an n^{th} order Markov chain over both words or large n-grams is presented. First, the training text is preprocessed to determine the words or n-grams present. As long as the system is not out of memory, the model stores words or n-grams and assigns random codes to them.

Next, combinations of three words are taught to the neural network. Suppose a statistical method: then one needs either (number of words) * 3 memory elements or sophisticated count, order, normalization, generalization and association methods [Brown et al., 1990b],

1. The results for the Markov chain over keywords were about the same as the ones over large n-grams. The only difference was in the need to define a dictionary in advance. Therefore these two algorithms are described in the same section.

[Jelinek, 1989 & 1991a-c]. The map shown below is a small part of a large 15 x 15 feature map.

INTERESTS-DISARM-PARTY	IMPROV-INTERN-CLIMATE	ATMOSPHER-NUCLEAR-STANDOFF
FAVOR-OUR-TALK	DON-ARM-RAC	AGREEMENT-ATMOSPHER-NUCLEAR
DISARM-ACHIEV-PEAC	WEAPON-DEVELOPMENT-DENOUNC	TRU-DYNAMISM-TALK
DENOUNC-REL-PRINCIPL	INTERN-POLITIC-SOVIET	AGREEMENT-NONDEPLOY-MENT-SPAC
FORTH-WID-RANG-EQUIPP	EVERYTH-NECESSARY	PARTY-TREATY-ENTIRELY
ENTIRELY-INTERN-POLITIC	STRENGTHEN-PROV-AGREE-MENT	ITSELF-DOUBLY-IMPOSS
PEAC-PROSPERITY-OBVIOUSY	SUPPORT-MAINTAIN-ABM	REDUCT-COMPLET-DESTRUCT
ARM-FORC-EQUIPP	TRU-HOP-N	WEST-SHOW-STRONG

FIGURE 17. Upper left part (3x8) of tri-words feature map of query in neural filter

To derive this map, the training text was preprocessed so all possible relevant words were determined. Two hundred eighty one words were found in the text on the Nuclear Weapons Restriction Talks. Then, the text was processed again and trigrams on words were determined and fed into the feature map.

5.3 Error Measurements in the Training Process

By measuring the error during the training process: $||w_r(t) - x(t)||$, an insight in the convergence properties of the neural network can be obtained. First, one has to understand that this neural network is used as a selection- and ordering device. Due to a smaller size than needed, only the most frequent n-grams are remembered (or trained properly), all others are forgotten, or overruled. Therefore, the average error will remain high (due to non frequent n-grams). In the first graph the total error in time is plotted (see next page). Globally, the error tends to decrease with the number of training cycles. The high errors on the right are infrequent trigrams that must be forgotten. (These are errors of n-grams which are continuously being bounced out).

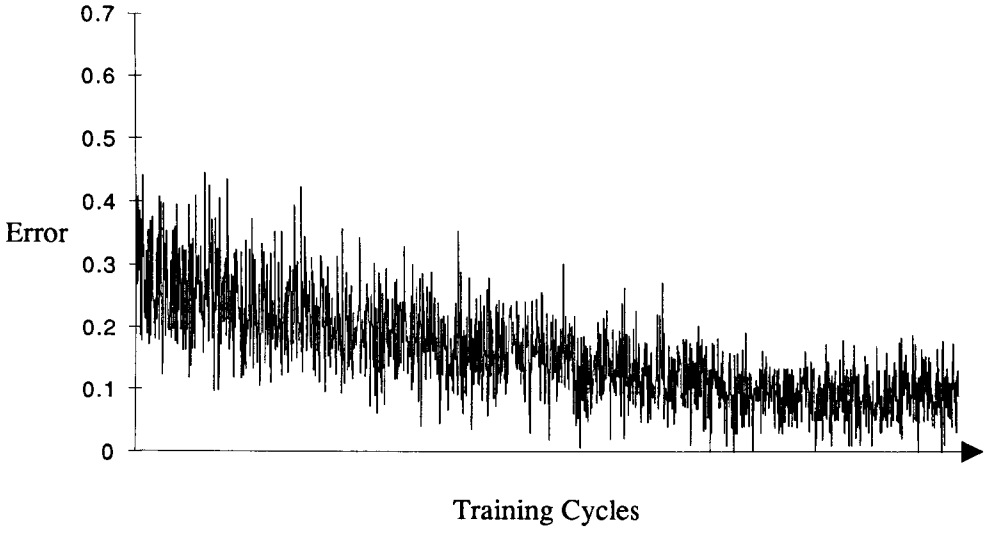


FIGURE 18. Error during training process

The next graph plots the error if it is smaller than 0.05. By plotting these bars, one sees that the frequency of almost perfect hits increases in time: the density of bars is much higher at the right side of the graph than at the left. This indicates that the model is getting better at representing n-grams.

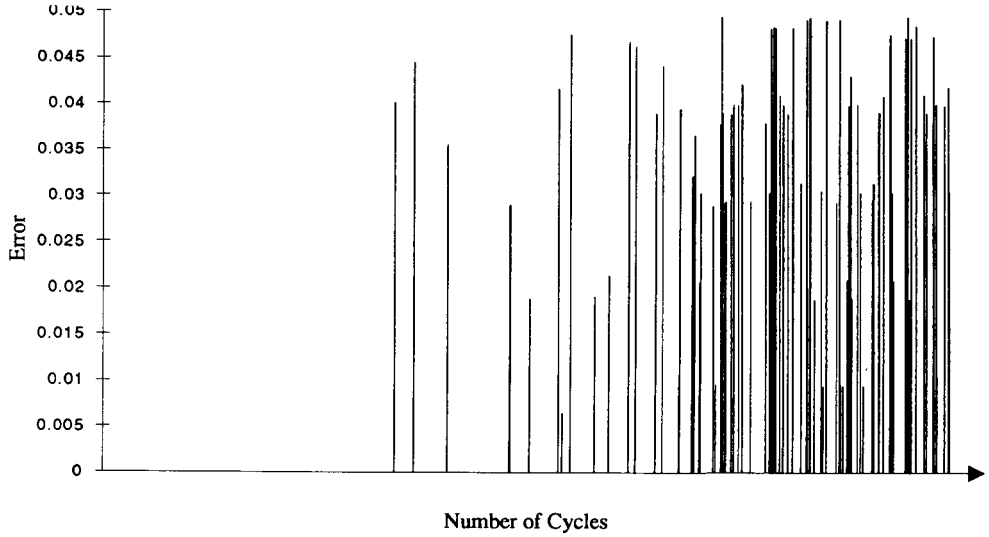


FIGURE 19. The perfect (<5% error) number of hits

The following graph represents the cumulative number of almost perfect hits (< threshold) in time. In the beginning, there are no hits at all. At a certain time, the number increases exponentially (self-organization starts). At the end, the number of hits stays constant (resulting in a linear increase of the cumulative value). These three graphs indicate that the network does indeed learns certain n-grams and it gets better at this task, the longer it trains, up to the moment the maximal capacity of the map is reached. From then on, only the most frequent ones are learned.

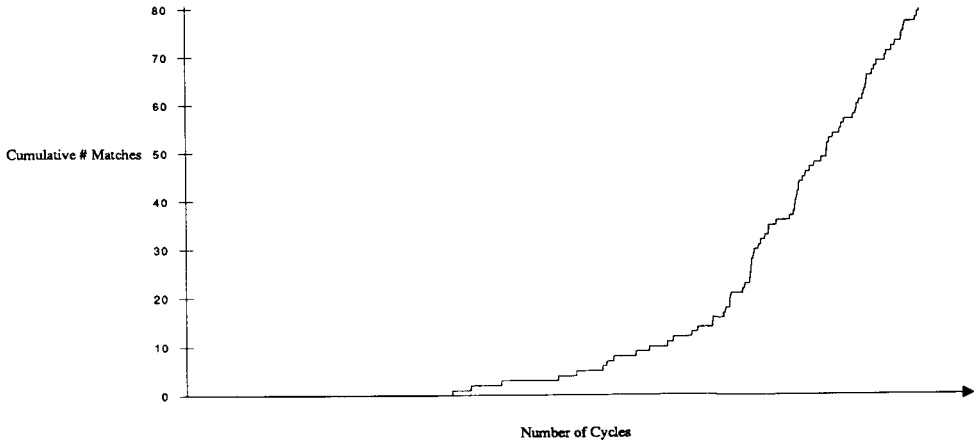


FIGURE 20. Cumulative number of perfect (<5% error) hits

5.4 Retrieval Results of the Neural Filter

5.4.1 Retrieval Results of the Neural Filter Based on N-Grams

The selection quality is a measure that cannot be given without being partial. In Chapter 2, the notions *precision* and *recall* are explained in more detail. There, it is argued that these values are very subjective and only relevant if compared to other techniques for the same data base and the same queries. However, even then, one might question the reliability of these numbers. Nevertheless, here some precision and recall values are calculated in order to make a comparison possible.

The precision and recall distribution given some parameter for one of the most advanced statistical Information Retrieval techniques [Croft et al., 1991], can be found in the next figure.

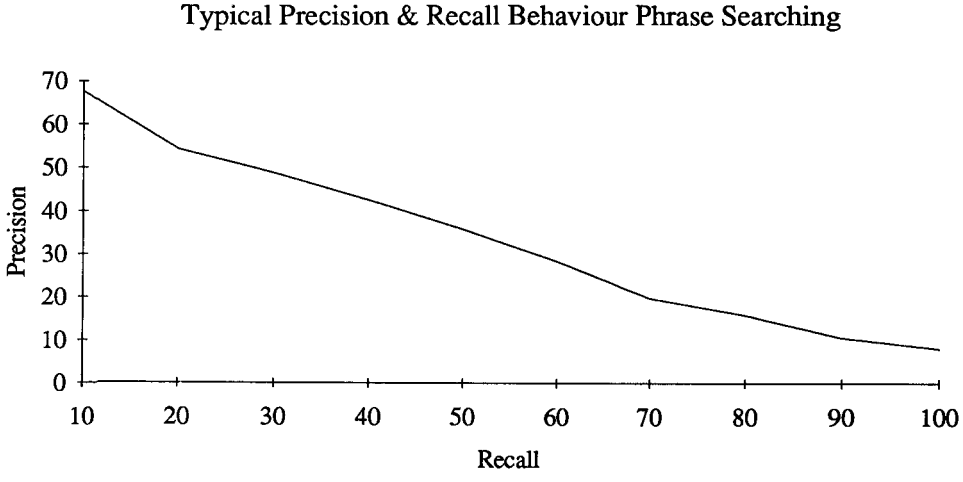


FIGURE 21. Typical precision and recall distribution for a phrase search in a statistical information retrieval system.

One can clearly see that the precision and recall are never both higher than 45%. If one of the two is set higher (by some parameter change), the other decreases dramatically.

On the next page, an overview of the results for the 4-, 5- and 6-grams is given. Per paper, a short description of the contents can be found. On the right hand, retrieval values are

given. The smaller the values, the higher the correspondence. The retrieval phase uses several different functions. The proposed function (average error per n-gram in the retrieval phase) separates related text parts clearly from non-related. Yet, the differences are quit small. That's what makes the Pravda interesting as a corpus[1]. There is much noise from words like comrade, socialism, hero, etc., making the retrieval phase more difficult (these words were not eliminated, but should have been).

By counting the average error per n-gram as well as the number of perfect hits, a better discrimination function is found. Generalizations caused by both the n-gram formalism and the Kohonen feature maps could be observed during the retrieval phase. The retrieval values (see following page) can be plotted in a graph. The lines represent (from lower to upper part of graph) the retrieval values for 50 text parts of the Pravda of the 4-gram, 5-gram, 6-gram and 8-gram analyses. Low values indicate low errors and thus high correlation. The first article is the same as the training text (because not all n-grams are taught to the neural map, a number of errors remains) It might be clear that the separation becomes better as the window size gets larger. This graph is based on the first selection rule: the normalized total error per text part. This is in fact a very negative approach (see Figure 22, "Retrieval results of the negative filter for the first 50 documents of the data base," on page 175).

Next, the precision and recall values are calculated for the 6 gram case. The variable in the calculation is the retrieval threshold value τ of Section 4.2.1 on page 158. Different precision and recall values can be given for different values of this threshold. In Table 10, "Retrieval values for negative selection (50 first papers from Pravda)", the articles related to the query are indicated with a "*". The precision and recall are calculated on basis of this manually made selection.

1. Others say the Pravda is a very bad corpus for information retrieval research because the language used involves many meaningfulness slogans: propaganda. However, we prefer the alternative view which states that the Pravda has a lot of noise, making it boring to read and therefore a useful application of IR techniques.

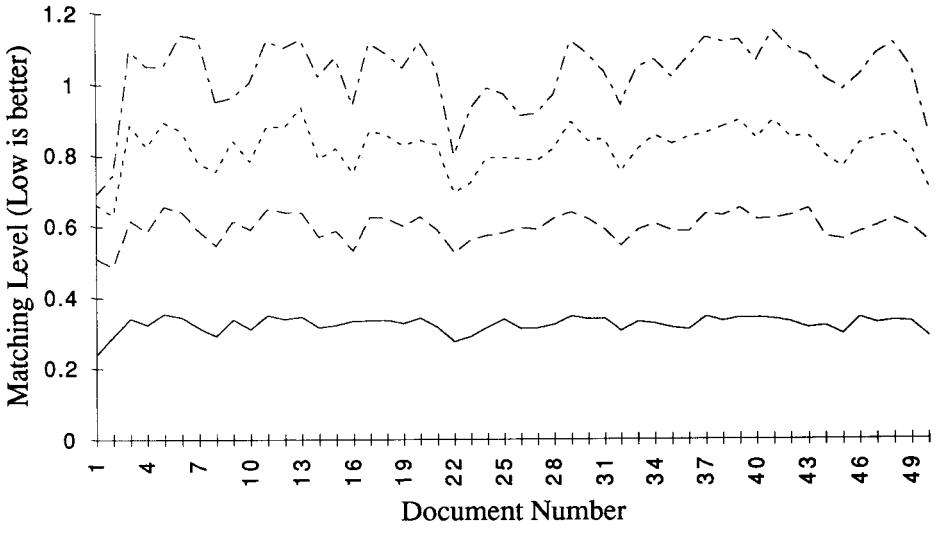


FIGURE 22. Retrieval results of the negative filter for the first 50 documents of the data base

TABLE 9. Precision and recall for negative filter on characters (7-grams)

Threshold	Precision	Recall
0.70	12.50	100.00
0.75	12.50	50.00
0.80	25.00	66.67
0.85	25.00	66.67
0.90	37.50	60.00
1.00	87.50	43.75
1.05	87.50	26.92
1.10	87.50	18.92
1.15	100.00	16.00
1.20	100.00	16.00

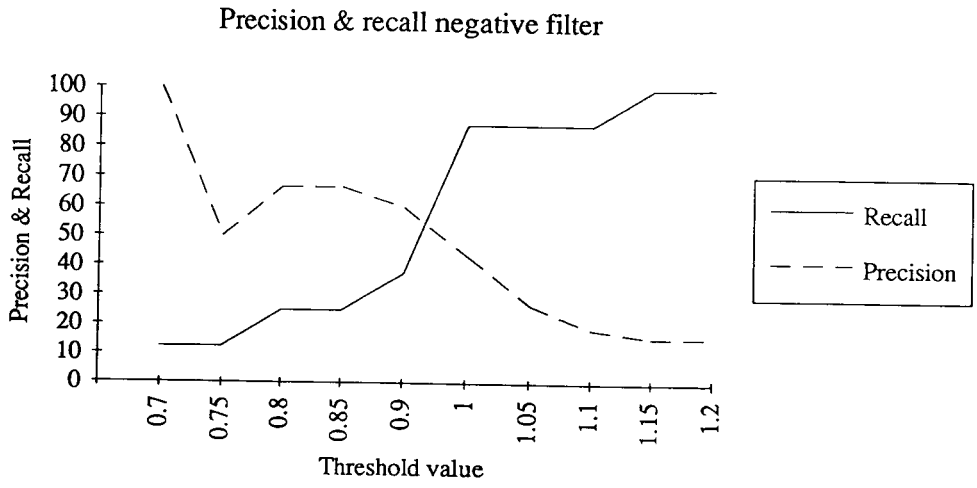


FIGURE 23. Precision and recall negative filter (7-grams)

The higher the precision as well as the recall value, the better the model. If these two values are high for a large range of values of the threshold τ, then the model is even better, because it does not depend too much on a proper chosen threshold value.

In the case of the negative filter, the results are not that good. They depend strongly on a proper chosen threshold. In addition, they are not really better than the standard values for a statistical method, as presented in Figure 21, "Typical precision and recall distribution for a phrase search in a statistical information retrieval system.," on page 173.

TABLE 10. Retrieval values for negative selection (50 first papers from Pravda)

Paper Nr.	Subject (* means somehow related to the query)	4-Gram	5-Gram	6-Gram	7-Gram
0	Weapons talk & socialist medals*	0.237318	0.509591	0.662473	0.689539
1	New years wish of M. Gorbachov	0.289728	0.488024	0.632248	0.744335
2	A day in the life of a museum director	0.341379	0.617132	0.884395	1.097080
3	Soviet Literature	0.322802	0.585385	0.823094	1.052009
4	A day in the life of ...	0.354099	0.657095	0.893633	1.052009
5	A 2nd world war hero	0.343971	0.640814	0.868803	1.139508
6	Nuclear weapon talks*	0.313828	0.586899	0.777539	1.128579
7	Peace demonstrations*	0.290589	0.546081	0.755270	0.951315
8	New years wishes	0.338634	0.613795	0.840996	0.961775
9	World news	0.310629	0.590616	0.782714	1.003833
10	A fairy tale	0.349794	0.647462	0.882928	1.122699
11	On rabbits	0.338259	0.639136	0.878360	1.101719
12	Poem on carnival	0.345716	0.638410	0.934767	1.126669
13	Central committee new years wishes	0.314312	0.569742	0.787559	1.023176
14	Labor news	0.321815	0.586252	0.818649	1.075319
15	On construction in the USSR	0.332173	0.530753	0.748884	0.940438
16	On transportation affairs	0.333690	0.626034	0.869077	1.116766
17	(Communist) party life	0.334771	0.621255	0.855764	1.085567
18	Space lift-off	0.325455	0.599938	0.825772	1.046907
19	Poem on nature	0.340744	0.625286	0.841849	1.118814
20	Story on a smoking teacher	0.314774	0.588360	0.828015	1.041139
21	USSR on foreign media*	0.273380	0.524630	0.692393	0.799374
22	Nuclear weapons talk*	0.288171	0.559702	0.720814	0.926369
23	On Afghanistan	0.313803	0.573072	0.792759	0.986893
24	On African countries	0.336627	0.579585	0.790675	0.969926
25	Economy: USA and EEC	0.309720	0.593978	0.788340	0.908930
26	Peoples Dreams	0.311645	0.588900	0.783028	0.915919
27	Satire on US Military*	0.322035	0.619409	0.815802	0.968615
28	Story on Italy	0.345008	0.637252	0.892476	1.120427
29	Carnival	0.337170	0.618419	0.838955	1.085120
30	Driving a car in the USSR	0.338137	0.588304	0.844606	1.032986
31	The party's social policy	0.302633	0.542491	0.750042	0.939680
32	Economy news	0.329482	0.587417	0.814469	1.045905
33	Around the world news	0.323436	0.605049	0.853771	1.068712
34	Work circumstances	0.312763	0.585953	0.830122	1.019806
35	Automation in baking industry	0.307923	0.583181	0.848865	1.076185

Paper Nr.	Subject (* means somehow related to the query)	4-Gram	5-Gram	6-Gram	7-Gram
36	Nature	0.344264	0.634208	0.859321	1.130220
37	The good old past	0.331045	0.628162	0.877461	1.117062
38	Theatre	0.341568	0.649438	0.896636	1.123436
39	Cambodia	0.341709	0.617094	0.844380	1.064060
40	Life in France	0.337273	0.619542	0.895881	1.149020
41	A letter to Santa	0.328580	0.628589	0.845106	1.096221
42	Sports		0.312252	0.646526	0.852348
43	Industrial reports	0.316586	0.568280	0.795445	1.011926
44	Nuclear weapons talks*	0.293399	0.560232	0.762709	0.981953
45	Gasoline	0.339407	0.579052	0.828988	1.021601
46	Product quality	0.324115	0.596635	0.842714	1.082056
47	On justice	0.331254	0.615968	0.857014	1.112873
48	On genetics	0.327425	0.593382	0.815307	1.043186
49	Nuclear Weapons*	0.286680	0.552238	0.706248	0.859771

If one uses the second retrieval function (a more positive one), the results are even better (see next graph). By counting the number of (almost) perfect hits and comparing the normalized value with a threshold (perfect retrieval is 100% in graph), the 7-gram training text has a 90% retrieval value[1]. Even a small paragraph mentioning the subject results in an already high peak in the graph.

1. This is quite high because the original learning text contains more words than the neural map can store. Therefore not all n-grams are remembered. Exactly these n-grams are responsible for the retrieval error. If one takes a negative approach (the first retrieval function), this percentage will be much larger than in the case of a positive one. This is why the second retrieval function works better.

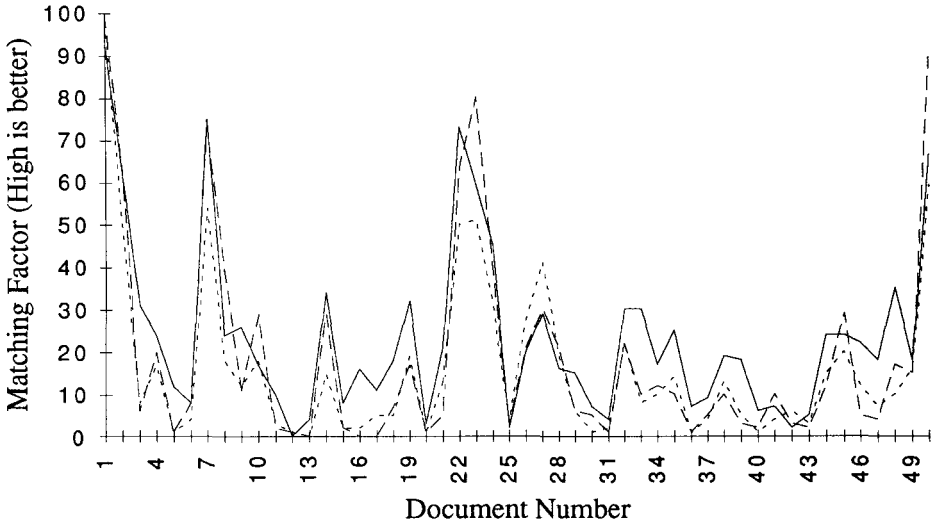


FIGURE 24. Results of the positive neural filter for the first 50 documents in the data base

Not all papers of the Pravda have been scanned by hand. Even if this had been done, it would be really difficult to express the amount of correlation in meaning. In many cases, it is quite easy to interpret the results in a different way[1]. Therefore, some related as well as some unrelated articles were inserted randomly in the test set. The model found them all with the proper retrieval values. Besides the inserted articles, all other articles found related to the query were in fact on the nuclear weapons talks between the USA and the USSR and not on conventional weapons, Chernobyl, other nuclear power plant, etc.

The determination of the most efficient retrieval function is a domain for study in itself. Obvious experiments can be done about combining a negative and positive training rule. More mathematically based correlation functions can be incorporated, etc. This is a main topic of future research. Pointers can be found in the literature on statistical pattern recog-

1. A standard IR evaluation technique compares the documents selected by hand with the documents selected by the computer. Here such experiments are not carried out due to the large amount of time required, but future research does not exclude this method of evaluation. Moreover, a standard benchmark data collection would be very interesting. The author is not aware of the existence of one.

nition [Sammon, 1969], [Duda et al., 1973], [Small et al., 1974], [Fu, 1977], [Croft, 1977, 1980, 1981], [Bokhari, 1981], [Devijver et al., 1982], [Voorhees, 1985], [Siedlecki, 1988].

In this case too, the precision and recall values, given the document set of Table 10,

TABLE 11. Precision & recall positive filter

	7-gram	7-gram	6-gram	6-gram	5-gram	5-gram
Threshold	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**
5.00	100.00	18.60	100.00	26.67	100.00	25.81
10.00	100.00	18.60	100.00	36.36	100.00	33.33
15.00	100.00	22.60	100.00	44.44	100.00	47.06
20.00	87.50	31.82	100.00	53.33	62.50	50.00
25.00	62.50	33.33	87.50	58.33	62.50	55.56
30.00	62.50	33.33	75.00	75.00	62.50	62.50
35.00	62.50	71.43	75.00	75.00	62.50	71.43
40.00	62.50	71.43	62.50	83.33	62.50	71.43
45.00	62.50	83.33	62.50	83.33	62.50	83.33
50.00	62.50	83.33	62.50	83.33	50.00	100.00
55.00	62.50	83.33	62.50	83.33	25.00	100.00
60.00	50.00	80.00	62.50	83.33	12.50	100.00
65.00	50.00	100.00	50.00	100.00		
70.00	37.50	100.00	50.00	100.00		
75.00	12.50	100.00	37.50	100.00		
80.00	12.50	100.00	25.00	100.00		
85.00	12.50	100.00	11.00	100.00		
90.00	12.50	100.00	12.50	100.00		
95.00			12.50	100.00		
100.00						

"Retrieval values for negative selection (50 first papers from Pravda)", can be calculated.

The precision and recall values of the positive filter are much better than those of the negative filter. Here, the precision as well as the recall are high for a large range of threshold values (see figure 25, "Precision & recall positive filter (7-grams)").

In addition, the results obtained for the Neural Filter are much better than the statistical values from Figure 21, "Typical precision and recall distribution for a phrase search in a statistical information retrieval system.," on page 173.

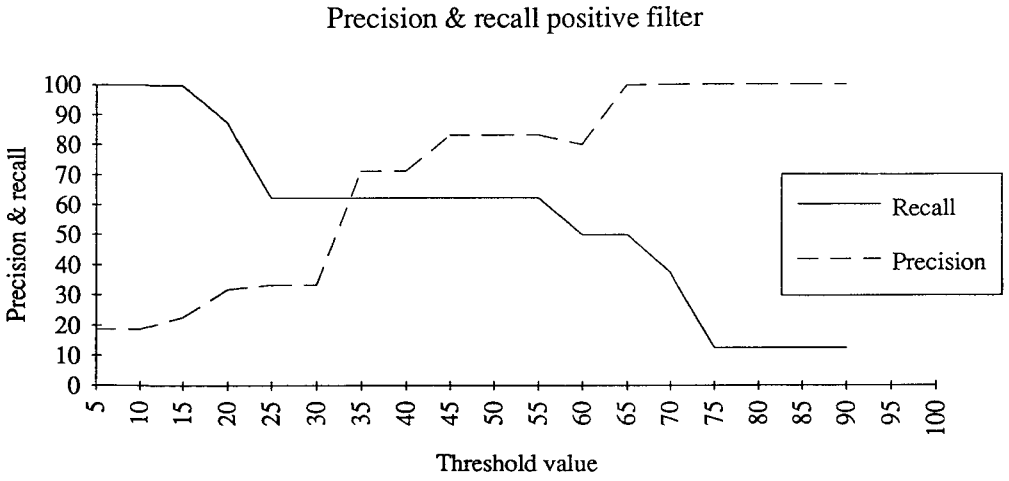


FIGURE 25. Precision & recall positive filter (7-grams)

5.4.2 Retrieval Results of the Neural Filter Based on N-Grams with Spaces

Similar simulations as above were done on the neural filter map trained with characters as well as spaces. By doing so, the map is also capable of expressing a correlation between word relations in the training and test text.

The number of possible n-grams held by the text increases dramatically as one incorporates spaces (without spaces a word of m characters ($m>n$) holds $m-n$ different n-grams, with spaces a word of m characters ($m>n$ or $m<=n$) holds $m+n-1$ different n-grams. Therefore, the size of the map must be larger compared to the case without spaces to remember the same amount of relevant n-grams. If the map is too small, too many relevant n-grams will be bounced out.

In the case with spaces, the number of perfect hits is much smaller than in the case without spaces. At first glance this might look discouraging, but a second look shows that, the difference between the more and the less correlated papers is larger than before. Therefore, selection thresholds are easier to set. The documents which are exactly on the same subject (and not just marginally correlated) have a very high relative number of hits. Ones which are only slightly related have a much lower retrieval value.

5.4.3 Retrieval Results of the Neural Filter Based on Keywords or Dynamically Chosen N-Grams

The retrieval of n-grams on keywords has one tremendous advantage over the n-gram on characters: it is incredibly fast (40 MBytes Pravda in 5 hours on a PC). Because the system only knows 281 keywords (all the non-trivial words in the training text), all other words do not have to be fed into the feature map, they are ignored. If on the other hand, proper word combinations are encountered, their retrieval value can be examined by feeding the word n-gram to the feature map. Due to the small number of known keyword combinations, the threshold for a perfect match must not be too high.

Sometimes, this method filters too much, but if one really wants only the most correlated objects from a large amount of data, this method can do so. Moreover, the same holds here as with the n-gram character filter with spaces: it should be used in combination with other, less strong filters to achieve a high quality filtering mechanism.

5.5 Complexity of the Neural Filter

During the experiments, the model showed linear behavior with respect to the window size. Here, the experimentally derived results will be proven correct by determining the complexity of the algorithm of the neural filter. The complexity is determined on the basis of a serial implementation. The fact that the neuronal algorithm can be parallelised easily is ignored completely.

TABLE 12. Used symbols

Symbol	Description
N	Number of elements (possible occurrences)
n	Order of Markov chain (window size)
T	Number of n-grams in training text
s	Number of sensors per window
r	Number of neurons in update region
L	Learning factor (the number of times the training set is passed)
p	Number of most frequent n-grams
C_m	Memory complexity (space)
C_c	Computational complexity (time)

The algorithm filters the strings before they are processed. This is not interesting for a complexity analysis (except if it was the most expensive step in the calculation, which it is not here). The following steps are important for the complexity: the neural algorithm has two phases, a training phase and a retrieval phase.

Two cases are distinguished:

- All n-grams are calculated, ordered and normalized. In the retrieval phase all n-grams are used.
- Only the p best n-grams are selected from the entire n-gram vector. These elements are normalized and used in the retrieval phase.

The reason for this separation might be clear. In the first case, many elements equal zero. Therefore, a calculation based on the p best elements of the n-gram vector probably evolves to the same selection outcome. By using only the p best elements, the complexity can be reduced considerably.

If all n-grams must be ordered, a number of neurons between q and N^n must be used, where q is the number of n-gram elements unequal zero. For large texts, this value reaches N^n, therefore, N^n neurons are used in the determination of the complexity. In the training phase, the number of calculations equals the times the training set is passed, multiplied by the number of trigrams, multiplied by the number of calculations needed to learn one n-gram. This last term is determined by adding the calculations for the determination of the best element on the map and the update of the weights within one region. To compute the best neuron, $s \cdot n$ sensors of N^n neurons must be evaluated. To update the weights, $s \cdot n$ sensors of r neurons must be updated. So for the training phase the following time complexity holds:

$$C_c = L \cdot T \cdot N^n \cdot s \cdot n + L \cdot T \cdot r \cdot s \cdot n \Rightarrow C_c = O(x^n \cdot n) \qquad \text{(EQ 17)}$$

The complexity of the training process is $O(x^n.n)$. In the retrieval phase, the update term r and the number of times the training set passed the training procedure are eliminated, resulting in fewer calculations, but with the same complexity. The time complexity for the retrieval phase is:

$$C_c = T \cdot N^n \cdot s \cdot n \Rightarrow C_c = O(x^n \cdot n) \qquad \text{(EQ 18)}$$

The number of memory cells needed is large. There ought to be a neuron for every possible n-gram. The memory complexity for the training as well as the retrieval phase is:

$$C_m = N^n \cdot s \cdot n \Rightarrow C_m = O(x^n \cdot n) \qquad \text{(EQ 19)}$$

Because the bigger part of the n-grams equals zero, it is quite silly to use all n-grams. The ordering phase is responsible for the largest part of the calculations. Therefore, this is the place to optimize the algorithms. By extracting the p best n-grams, the number of calculations in the ordering phase is reduced.

However, in the neural algorithm, the number of n-grams used is determined by the number of neurons in the map (every neuron can hold up to one n-gram). All less frequent n-grams are absorbed by the more frequent ones. If p neurons are used, the number of calculations needed in the training phase is dramatically reduced. The time complexity for the training phase is then:

$$C_c = L \cdot T \cdot p \cdot s \cdot n + r \cdot s \cdot n \Rightarrow C_c = O(n) \qquad \text{(EQ 20)}$$

The time complexity for the retrieval phase is:

$$C_c = T \cdot p \cdot s \cdot n \Rightarrow C_c = O(n) \qquad \text{(EQ 21)}$$

The memory complexity for both cases is:

$$C_m = p \cdot s \cdot n \Rightarrow C_m = O(n) \qquad \text{(EQ 22)}$$

So, in the case of a restricted sort (which is more than enough as the exhaustive sort is an absolute overkill) the neural filter has a *linear* complexity in time and space with respect to the window size [1].

Please be aware that the neural complexity calculations are based on serial simulations and not on parallel ones. If the Kohonen feature maps were to be implemented in large neural chips, the results would be even better[2].

1. In the calculations, it is assumed that larger window sizes do *not* yield longer training times. This expectation is derived from the experience obtained during the simulations. However, if this is not the case, then there is an increase by a factor in which there is no insight yet.

2. Moreover, using methods proposed by [Kelly, 1991] and [Koikkalainen et al., 1989] the search for the Best Matching Unit (BMU) can be done with the aid of a binary tree, storing the weight vectors in an ordered fashion. Then, the complexity in speed decreases from $O(n)$ to $O(ln\ n)$. The complexity in space increases from $O(n)$ to $O(n.\ ln\ n)$.

5.5.1 Scalability of the Neural Filter

An important question that has to be answered concerns the measure of scalability of the algorithm. How well can one extend this algorithm to larger data sizes. A first preliminary simulation that is carried out passed 3 MByte of Pravda along a 15 by 30 neuron feature map. A window of size 4 was used. After three passes of the entire text along the neural filter, the Euclidean distance between all neuron vectors was about zero, indicating that all neurons represent text parts that are related to the text parts represented by their neurons(-see figure 26). However, training took *very* long and the model has to be trained by using the hypermap algorithm. The regular Kohonen feature map did not work so well. So scalability can be considered a weak feature of the model.

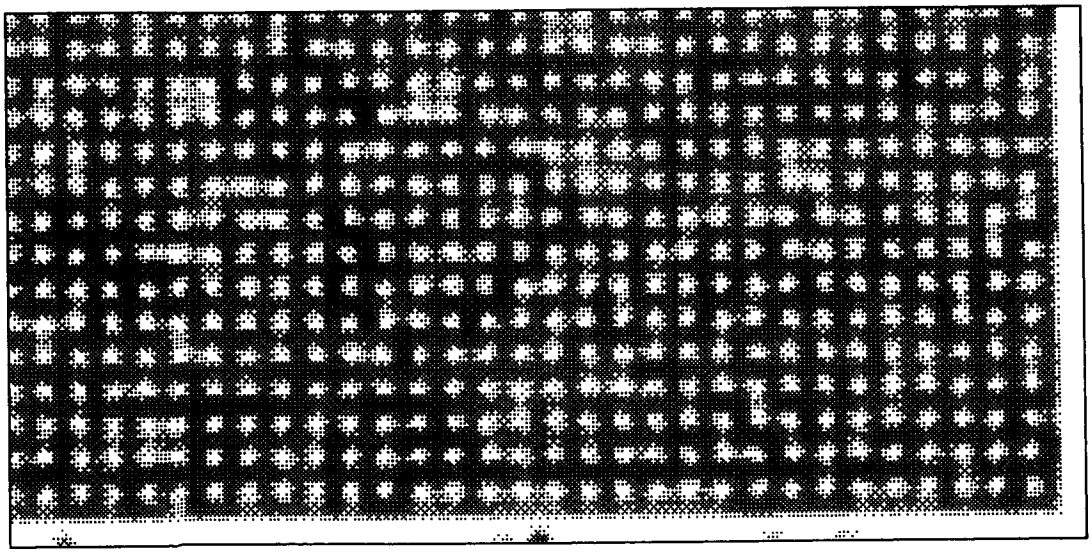


FIGURE 26. Euclidean distances between neurons on the feature map. The white dots indicate the neurons. High Euclidean distances yield a dark color. Small distances yield a gray color. Most distances are represented by gray colors, indicating that all values are closely related to those represented by the neighboring neurons.

5.6 Simulations and Results Neural Interest Map

A vector representing text distribution features can be derived for every paper in the data base. Such a vector then represents a fingerprint of a data base object. Fifty papers were scanned and their corresponding vectors were taught to the neural network in the following simulations. The Kohonen training mechanism was just standard. No special features were used. Two aspects should be pointed out before the simulations are discussed. First, these simulations differed from the ones proposed by [Gersho, 1990a-b], [Wermter, 1991] and [Lin et al., 1991] in that the former used very restricted text parts for the derivation of the feature vectors (mostly titles) and that the methods were based on a well optimized hand-made keyword selection. Here, the keywords were derived from the text in the objects automatically and the feature vectors are based on keyword distributions in the training text. In other words, the model as presented here implemented a *full-text* model. Therefore, the vectors had very high dimensions and required long training times. The result was a fully automatic clustering mechanism.

The Neural Filter simulations could still be implemented on a high end PC. These Neural Interest Maps required a more powerful computational basis. This was due to several reasons. First the simulations needed many more training cycles. These training cycles on their own took longer because they were based on larger vectors (500 to 2500 dimensional). However, the main reason why the PC was no longer suited for the simulations was that the standard PC could not calculate with large enough precision to guarantee convergence. One definitely needed the extended precision calculations of the Sun IPC to organize the elements in the map based on very small differences in vector dimensions. Sometimes, even then the map could not converge. This was particularly the case with the n-gram based simulations.

5.6.1 Preprocessing Keywords and N-Grams

Before the vectors can be taught to the neural map, they have to be derived from the free-text data base in the first place. This can be done by some preprocessing programs. As an initial step one derives the m most frequent words used in all the text parts. Next, the word distribution of these m words in the n text parts must be calculated. The distribution can be expressed in various forms:

- The occurrence can be measured (0 equals no occurrence, 1 equals the keyword occurs, the number of times it occurs is ignored).
- The word frequency can be normalized with respect to the total number of keywords in all text parts.
- The word frequency can be normalized with respect to the maximum occurrence of this specific keyword in all the text parts.

- The word frequency can be normalized with respect to the total occurrence of this specific keyword in all the text parts.

Once the n vectors of m dimensions have been derived, they are trained to the neural network in a random way. After a certain training time, the neural network holds a representation of the relations between the papers in the data base. Related papers will be stored in neighboring neurons.

When one substitutes the keywords by n-grams (with spaces), a language independent clustering method results. First, the k most frequent n-grams are determined. Next, the distribution of these k n-grams in m text parts is examined. The obtained vectors can be normalized in the same different ways as the keywords are (see above). The n-gram vectors are much larger than the keyword vectors. In general, only those n-gram dimensions that were unequal to zero, were presented to the network. To increase performance, the number of n-grams can be reduced even more. However, this can limit the cluster information. The normalized n-gram distribution vectors (one per text part) are taught to the Kohonen feature map in random sequence. After training, the map holds related papers in neighboring areas.

As with many computational linguistic problems, the solution of the clustering problem lies in the proper choices of the data representation.

5.6.2 Results Interest Map Based on Keywords and Large Preselected N-Grams

In the simulations of the Neural Interest map based on keyword frequencies, a 10 by 8 or 10 by 10 map with 500 input sensors per neuron was used. The maps were trained somewhere between 5.000 up to 15.000 train cycles.The following maps were found:

0	1	0	2	xxx	23	xxx	7	9	11
0	0	4	3	xxx	1	35	12	xxx	xxx
xxx	xxx	xxx	20	20	47	47	35	36	32
21	21	22	20	xxx	34	xxx	45	45	28
21	26	xxx	xxx	44	39	39	45	31	32
27	25	xxx	24	xxx	39	39	31	31	31
29	xxx	19	xxx	30	xxx	49	41	38	48
29	15	xxx	xxx	xxx	42	40	xxx	46	38

FIGURE 27. 10 by 8 interest map of 500 keywords: normalized with respect to the total number of words

4	4	46	20	20	xxx	13	xxx	6	xxx
xxx	46	46	46	20	5	13	xxx	xxx	2
41	41	xxx	xxx	xxx	5	xxx	45	45	xxx
xxx	8	8	xxx	31	xxx	32	xxx	xxx	1
29	29	xxx	17	xxx	49	xxx	28	xxx	xxx
xxx	0	0	xxx	48	xxx	37	xxx	18	xxx
23	23	xxx	12	xxx	35	xxx	42	xxx	27
23	34	34	xxx	xxx	40	38	xxx	22	7
xxx	26	34	xxx	47	xxx	44	25	19	10
26	26	xxx	43	xxx	33	xxx	30	21	16

FIGURE 28. 10 by 10 interest map of 500 keywords: normalized with respect to the total occurrence of the word in a specific dimension

31	xxx	xxx	xxx	18	xxx	xxx	8	xxx	9
32	30	xxx	20	xxx	xxx	19	xxx	7	xxx
34	xxx	23	xxx	22	21	xxx	6	xxx	3
xxx	25	xxx	24	xxx	xxx	43	xxx	5	4
33	xxx	26	xxx	44	42	xxx	45	xxx	1
xxx	49	xxx	46	xxx	xxx	41	xxx	2	xxx
47	xxx	48	xxx	37	xxx	xxx	40	xxx	0
47	47	xxx	38	xxx	35	36	xxx	39	xxx

FIGURE 29. 10 by 8 interest map of 500 keywords: normalized with respect to maximum occurrence of a word in a specific dimension

Although the above three maps hold some interesting relations, the overall conclusion is that they are quite wrong. The papers 0, 1 and 49 are all on nuclear weapons restriction talks. Zero and 1 are in neighboring regions, but 49 never is. The papers 25, 32, 34 are all on economical issues, etc. The vectors used were too much related due to the small values caused by the normalizations used. The coding that did work well was the one where each dimension represented one word. If a word occurred in the text part, the dimension equals one, otherwise it equals zero. By eliminating the frequency of occurrence, the vectors became more distinguishable and therefore better trainable. The picture on the next page shows the map obtained after 15.000 training cycles.

20	xxx	22	21	24	xxx	48	48	xxx	47
xxx	18	xxx	19	xxx	46	xxx	xxx	49	xxx
23	xxx	35	xxx	38	37	xxx	40	xxx	0
23	xxx	xxx	33	xxx	xxx	39	xxx	41	xxx
xxx	36	xxx	xxx	xxx	34	xxx	42	xxx	1
2	xxx	12	11	10	xxx	44	xxx	43	xxx
3	xxx	13	14	xxx	17	xxx	45	xxx	32
xxx	4	xxx	15	xxx	xxx	16	xxx	31	xxx
5	xxx	7	xxx	xxx	26	xxx	29	xxx	30
xxx	6	9	8	xxx	xxx	25	27	28	xxx

FIGURE 30. 10 by 10 interest map of 500 keywords. 01 vectors; 0: word does not occur, 1 word occurs

There are two clusters related to war. One on the bottom left which holds documents on conventional warfare. The upper left cluster is based on the more scientific SDI warfare. Why these two clusters are so far separated is not clear. On the right side, a large cluster with economical documents can be found, Within this group, smaller neighborhoods hold a transportation cluster, socialism clusters and product quality (a hot item in the USSR) clusters. The more manual labor based economical documents are on the left side. Surrounded by a cluster for Art and one for Nature (e.g., Africa). This can be translated to the following global area found in Figure 31, "Areas found on the feature map and related topics," on page 190.

Although some groups are still separated for reasons which are not yet clear, the overall impression is that this map holds many of the semantical relations between the documents. However, much of the relations are not correct.

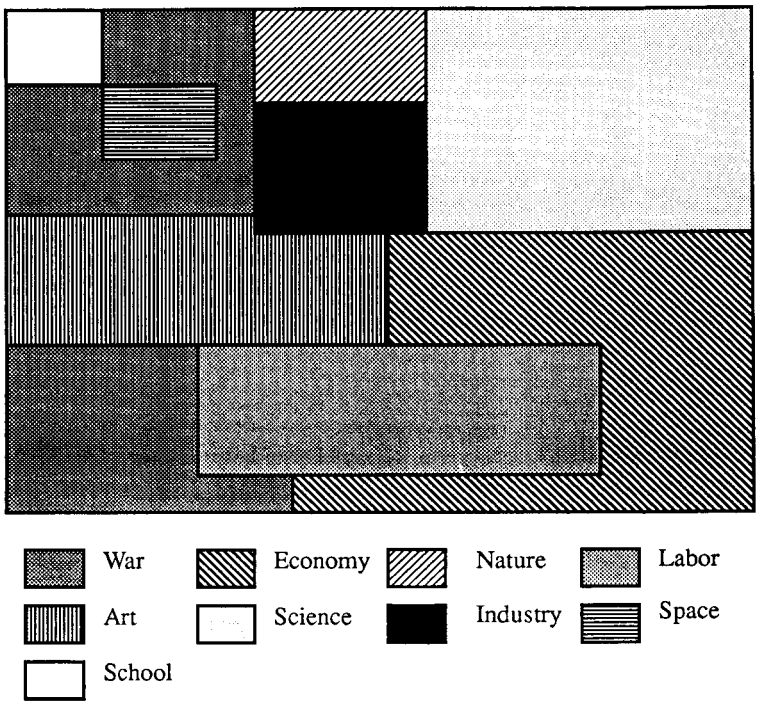


FIGURE 31. Areas found on the feature map and related topics

5.6.3 Results Interest Map Based on Trigrams

The words in the simulations just mentioned were derived automatically, so the step to n-grams is not that large. By incorporating spaces, relations between words can be characterized by n-grams. Because the system cannot learn too many n-grams, the m most frequent n-grams can be taught only. This set of n-gram vectors must be derived first, resulting in an exponentially complex problem with respect to the size of n. Therefore, only simulations for $n = 3$ were carried out. Because the differences between the vectors were very small and the vectors very large, one cannot use the actual frequencies of trigram occurrences in the training set. Therefore, 0 and 1's were substituted for the actual frequencies, as in the simulations above. The map formed organized itself after long training times. The results are comparable to the keyword based organizations, although it takes much longer to derive them.

5.6.4 Retrieval Neural Interest Map

The retrieval is always based on the calculation of a keyword or n-gram vector with classical methods. The keyword match is quite fast, because one only needs to calculate the frequency of known words. Normally, the dictionary is of restricted size. By incorporating advanced hashing techniques, the elements can be updated quickly.

More difficult is the derivation of an n-gram vector per document. Although one also has a restricted feature space, it is much larger than the keyword space. Moreover, search times are longer for n-grams than for keywords (see Neural Filter results for a motivation).

Once the vector has been determined, it can be fed forward to the feature map to derive the Best matching Unit (BMU). This BMU represents the document most correlated with the vector. The neurons within a certain euclidean distance *d* hold related documents. However, one needs the BMU as well as the cluster boundaries to make a responsible decision on the measure of correlation between documents (see figure 32, "An ideal feature map").

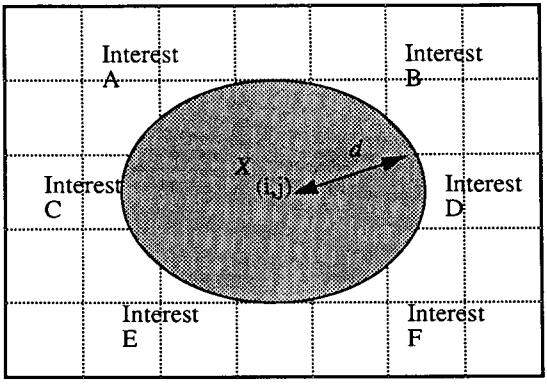


FIGURE 32. An ideal feature map

If the vector *x* correlates best with the BMU with weight *w* at neuron (i,j), then all neurons within distance *d* are supposed to be related (dark circle). But, what if the neuron at the feature map looks like the one derived in the Interest Map on the next page If the BMU seems to be the neuron at position (i,j), it is positioned exactly at the border of multiple interests. The reason why these interests are neighboring is not because they are related, but because they are forced to interconnect due to the dimension reduction properties of the feature maps. If one uses the euclidean distance as a selection criterion, the documents

selected are not the proper ones. Some interests which are neighboring have nothing to do with each other.

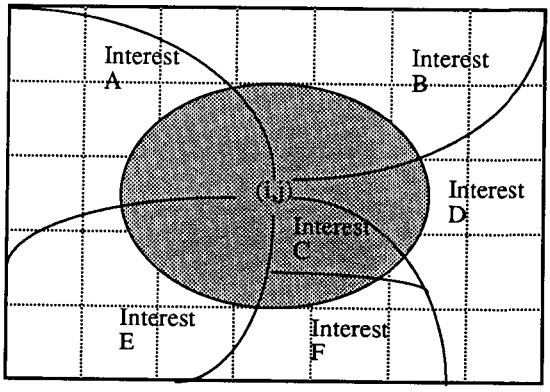


FIGURE 33. A feature map as they occur in the real world

So, the only way to make a reasonable decision is by incorporating the BMU as well as the cluster boundaries. This is a big disadvantage, because then one has to determine the cluster boundaries manually. These decisions take much work, are very personal and therefore subjective and sensitive to errors[1].

More on this problem can be found in the next sections.

1. Determination of cluster boundaries can also be done automatically by measuring the distance between the weights of two neighboring neurons (the Euclidean as well as the Cosine distance can be used, although the latter will be more suited), a feature map landscape can be calculated which holds high values for non-related neighboring neurons, also called fractions in the homogeneity of the distribution. These hills might then be incorporated as cluster boundaries in the selection process to avoid selection of non-related objects. However, the clusters obtained will never be as good as those derived manually. Therefore, the cluster determination remains an open question.

6.0 Discussion

This section discusses several items in the field of neural information retrieval. To get a better overview, the subjects are organized under different headings.

6.1 Neural Networks for Information Retrieval & Information Retrieval for Neural Networks

Information retrieval, being a clear pattern recognition problem, has mainly benefitted from statistical pattern recognition technologies. The enormous amount of data to be processed actually did not allow any other methods within practical limitations. As time passed, many researchers tried to increase the level of analysis without blowing up the computational needs. Due to this constraint, the information retrieval tool box could never use any linguistic theory. Therefore, the algorithms used are restricted to local surface analyses.

Recent research in connectionist natural language processing showed interesting self-organizing models that can learn approximations of finite state grammars and simple semantical relations from unformatted data. Moreover, these neural devices showed remarkable competence in clustering and classification tasks of incomplete data sets. All these properties are well known functional demands for information retrieval systems. Maybe that is the reason why the number of papers appearing in literature is increasing so rapidly [Belew, 1986], [Personnaz et al, 1986], [Doszkocs et al., 1990], [Gersho, 1990a-b], [Kwok, 1989, 1990,1991], [Rose, 1990], [Allen, 1991], [Wermter, 1991], [Lelu, 1989, 1991], [Hingston et al., 1990], [Wettler, 1989,1990], [Mozer, 1984, 1991], [Rapp et al., 1991a-b], [Bochereau Laurent et al., 1991], [Brachman et al., 1988], [Eichmann et al., 1991, 1992], [Jung et al., 1991], [Van Opdorp et al., 1991] [1].

Besides the positive contribution from neural networks to information retrieval, there is also one the other way around. Information retrieval has a long and well understood history in statistical pattern recognition. Many problems have indeed been solved by using statistical methods. Comparisons of such results with new results in neural information retrieval open doors to a better insight in the exact relation between neural networks and other classical pattern recognition solutions. Because, if neural networks are such good pattern classifiers, where does one position it with respect to the known pattern recognition theories? (See Chapter 6 for a more detailed discussion on this topic).

Even more interesting is the contribution of information retrieval to NLP. Because information retrieval problems are often much simpler, they clarify neural bottle-necks much

1. See also Chapter 2: "Neural Networks in Natural Language Processing and Information Retrieval" for an overview of applications of neural networks in Information Retrieval.

more easily than NLP problems, thus contributing to the development of better neural models for NLP.

6.2 Neural Information Retrieval versus Traditional Information Retrieval

Throughout this chapter well known notions like inverted files, indexes, relevance ranking, relevance feedback, precision and recall have not been used at all. So what is the relation to those established techniques in information retrieval?

First, the reason why these notions have not been referred to is that all of them apply to the retrieval of bibliographic data bases in the first place. The focus of this chapter has been on the real-time filtering problem, which can also be defined in therm of data base retrieval, but which has some significantly different characteristics. Second, as it is argued, the manner in which the precision and recall measurements are used could still be doubted.

However, as inverted files, relevance ranking, relevance feedback and the addition of thesauri are known achievers in information retrieval, their neural counterparts in the neural filter as well as in the neural interest map shall be pointed out.

6.2.1 The Neural Filter as a Traditional Information Retrieval Device

- *Inverted Files* cannot be found in the neural filter, because the neural filter derives an internal representation of the query and not of the data base. Therefore there are no tables with keywords to file names. However, the neural network does contain words as they occur in the query. One or more neurons can represent such a word.
- *Boolean Queries* cannot be implemented directly in the neural filter. However, they can be in the external shell. The neural network is then used as an associative memory for the query engine.
- *Relevance Ranking* is something that can be implemented through the external shell. Again, the neural network can be used as an associative memory for the words used in the query. The values obtained from the neural network are better than the probabilities used in traditional information retrieval as they are conditional probabilities with respect to contextual structures in the document as well as to the entire data base. Not only do these values indicate the relative frequency of an object, but also the context in which they occur. Traditional relevance ranking is only with respect to the number of occurrences of objects in the document and in the entire data base.
- *Relevance Feedback* is something which can be implemented easily in the external shell. There are many possibilities for relevance feedback in the neural model. One can use the topological formations on the feature map, or one can include properly retrieved documents in the query easily.

- *Thesauri* can be implemented in a neural as well as in a traditional manner. As neural networks are well understood associative memories, their application in a thesaurus might be clear. The only problem one may encounter are the contradictions and the measure of relation between different words.
- *Latent Semantic Indexing* can be implemented in the external shell as well as in the neural network. In the latter case, groups of words clustered in a neural network might be used as input translator for the neural filter. This means that all words are first mapped onto the semantotopic map. Next, the area coordinates of the region which is most activated is used as input for the neural filter.
- *Conceptual Information Retrieval* is a natural extension of Latent Semantic Indexing for the cases more than just simple semantically related groups are available. Besides these word clusters, more information on the concepts can be used by the model. More on this can be found in the next Chapter: "Neural Data Oriented Parsing".

6.2.2 The Neural Interest as a Traditional Information Retrieval Device

In fact, the Neural Interest Map implements Salton's vector space model [Salton et al., 1983], [Salton, 1989]. Therefore, a comparison is much easier than in the case of the neural filter.

- *Inverted Files* cannot be found in the neural interest map. Every document is represented by a keyword vector on the feature map. Relevant documents given a query can be looked up easily in the feature map by determining the BMU.
- *Boolean Queries* can be implemented in the external shell. There is no implicit mechanism in the neural network to facilitate this technique.
- *Relevance Ranking* is implicitly implemented in the euclidean distance between the input vector and the weight of the BMU. Here, the retrieval value is equal in quality as in the vector space model.
- *Relevance Feedback* can be implemented by investigating the documents represented by neurons in neighboring areas of the BMU. In addition, similar techniques as in the vector space model can be implemented by adjusting the query.
- For the *Thesauri, Latent Semantic Indexing*, and *Conceptual Information Retrieval*, the same comments hold as in the previous section.

6.3 Information Retrieval With N-Grams

A major drawback of a keyword-matching system is the need for a dictionary, and the lack of context. N-grams seem to provide an acceptable solution for a reasonable price. They are not sensitive to noise (see examples), they can be used without predefined linguistic knowledge, and they can be derived automatically. However, one should be aware of the

fact that they are more restricted than systems based on keywords or linguistic units (such as noun phrases).

By varying n between 3 and 7 a proper contextual analysis is derived without using dictionaries. If one incorporates spaces in such a higher order model, n-grams with starting and trailing spaces hold information on words transitions, thus implementing syntactical and (low level) semantical relations [Brown et al., 1990b], [Jelinek et al., 1989], [Jelinek, 1991a-c].

Here, the n-grams are used in the filtering as well as the clustering algorithm. In the filtering solutions they do particularly well. The ability to represent 7-grams without any computational pain definitely improves recognition rates. The n-gram over n-gram simulations showed even better results. On the other hand, the application of n-grams in the clustering algorithm was not that easy. It needed more computational power than expected. However, by using various optimization techniques, the results were derived more easily than if the simulations had to be hand-coded.

6.4 The Filter and Interest Map as Hashing Functions and Semantic-Cognitive Maps

If one studies the behavior of the *Neural Filter*, the question arises as to what its exact relation is with another well-understood addressing technology hashing [Boyer et al., 1977], [Bozinovic et al., 1982], [Harrison, 1971], [Knuth et al., 1977], [Larson, 1988], [McIllroy, 1982].

On the one hand, the relation is very clear: neural networks are large (calculating) associative memories, able to store elements efficiently. On the other hand, the reason why certain elements are stored and others are eliminated is not yet clear. More on this subject can be found in the following chapter.

The *Interest Map* stores various n-grams and vectors from a static text and matches this neural map against (dynamic) queries. Here, the use of neural networks does not have the same amount of success as the neural filter, although the latter works well as a generalizer. In general, the clustering of document spaces is known to be expensive ($O(n^2)$ complexity in time and space) and not useful. Therefore one might question the use of the results obtained for the neural interest map. However, this is the most obvious application of Kohonen feature maps in information retrieval, and therefore it has been investigated.

The relation between the Neural Interest map and better understood semantic and cognitive maps is obvious in a certain way. However, the Kohonen feature map is just a small step in the direction of the functional requirements as meant in literature. One should not overestimate the power of feature maps. In particular, the problems related to the cluster boundaries (due to the dimension reduction) are quite difficult to solve.

6.5 Higher Order Linguistics & Knowledge Representation

A major problem in information retrieval has been that higher level analyses were not available at reasonable prices. Recent research in connectionist neural networks showed how to teach approximations of finite state grammars to recurrent neural networks. By training word sequences, neural networks were able to learn regular grammars. Although these grammatical structures are the simplest possible, they can definitely increase recognition performance. However, the recurrent models learn very slowly and are quite unstable. But, it seems that most information retrieval systems are aided sufficiently with a restricted Markov model (such as trigrams over words). The fact that neural networks can learn these relations in linear time, opens up new possibilities for neural networks in information retrieval.

So much for the treatment of structural analysis in IR. Another important problem is to incorporate meaning in IR. Yet, there is no real meaning involved in the algorithms proposed. There are only the contextual relations incorporated in the n-gram representation. By generalizing over these contextual structures, simple semantic relations can be derived. However, real meaning and the interpretation of conceptual structures is something more complicated, and should not to be taken lightly or solved solely by means of n-grams.

Other research focuses on knowledge representation structures for information retrieval. The early connectionist models were mainly used for such applications. Only recently have neural networks been used for clustering tasks. A possible use of such clustering neural networks for knowledge representation is in the use of hierarchical feature maps, where relations between objects and classes of objects are captured in the hierarchical structures. Another solution can be found in [Allen, 1991], who uses a simple recurrent network (SRN) to teach semantical issues to a back-propagating network.

6.6 Problems with Kohonen Feature Maps

There are some serious problems with the feature maps as used in these simulations. First there is the neural filter. One does use this property of the feature map during the training phase, but it results in strange and sometimes unwanted effects. Frequent n-grams disappear and non-existing ones appear. However, if one eliminates the neighborhood effects and thus implement a form of Principle Component Analysis (PCA) [Oja et al., 1988, 1991], the in-frequent n-grams are no longer thrown out of the feature map[1]. If one reduces the neighborhood effects, the model converges more slowly and ends up repre-

1. According to Professor E. Oja, the underlying distribution function of the n-grams is much too clustered to use PCA's or comparable methods. Such mechanisms only work properly for very homogeneous data sets of noisy natural data.

senting the n-gram distribution less well[1]. So even if one does not explicitly use them (or does not exactly understand them), the neighborhood effects seem to have a significant role.

Next, how does one interpret a topological map of n-grams or keywords in the Neural Filter? This property of Kohonen feature maps is not used in the retrieval phase because one does not know how! This problem is closely related to some assumptions made about the underlying probability distribution. The Kohonen feature map requires a predefined network structure that should adapt to the underlying probability distribution (e.g., fixed dimension, fixed rectangular or hexagonal connection structure and fixed square, triangular, circular but continuous homogeneous topology). One of their main problems here, is that one does not know the form of the probability distribution of language. One thing is for sure, it is definitely not two-dimensional, rectangular and homogeneous distributed (as is presumed by the form of the feature map used).

This problem is even more clear in the case of the interest map. After the training phase, related objects must be in related neighborhoods. However, what if a paper is on the border of multiple clusters. If this neuron is selected as the Best Matching Unit (BMU) on the Kohonen feature map, then the Euclidean distance does not represent a proper measure of correlation. One has to incorporate the cluster boundary knowledge in the classification decision. Such cluster boundaries must be derived by the model itself and not by an external supervisor.

More on this topic can be found in the next chapter: "Neural Data Oriented Parsing" and in chapter 6: "Discussion and Conclusions".

6.7 Kohonen Feature Maps, Back-propagation and Other Neural Paradigms

Is the Kohonen feature map the best neural model for the simulations carried out? There are many other neural models. The early neural information retrieval used localist knowledge representations (one neuron for one concept). Recent efforts showed the application of feed-forward and recurrent back-propagating networks. Kohonen feature maps are just recently invoked in IR applications. Hopfield networks and other associative memories have also been used, but only rarely.

1. According to Professor T. Kohonen, it is really difficult to understand what is happening on the feature map. However, if the neighborhood function has been eliminated in other applications, training slowed down and the cluster boundaries on the feature maps were much more discontinuous. So, even if you do not understand the topological map, you can still use the neighborhood effects to end up with a smooth representation of the probability function of the training set.

In general, clustering and generalization problems are best solved with self-organizing networks, such as the Kohonen feature map, the Simple Recurrent Network (SRN), and ART. Mapping problems or function approximations can best be done by a feed-forward back-propagating neural network. Temporal processing can best be done by an SRN or any other recurrent model. Associative memory problems might be solved be either neural network: BP, Kohonen, ART, etc. Of course, these applications can also be solved with other network types, but the networks mentioned are the most natural choices.

Information Retrieval is a clustering problem. Based on a selection of specific features (e.g., n-grams or keywords), a representation of an object is derived by feature extraction. The objects are categorized in clusters by the retrieval function. The main issue is the determination of such features, so the difference between clusters is as big as possible (or, as little as overlap as possible, since overlap causes the classification error). Because the Kohonen feature maps are the computationally most effective self-organizing networks, they are in fact the best neural network for such problems. Once more, one does not want a supervised model, because it is not known what to learn in the first place.

However, it is also possible to use an SRN to teach a representation to the neural filter. A disadvantage of the application of an SRN in the neural filter is the fact that the model implements an high order Markov chain by using recurrent fibres. This is just much too sophisticated. If one uses a regular BP network with a window, the network does not form a representation as good as the Kohonen feature map. The representation is much more discontinuous.

Moreover, it is difficult to structure the input set. In the case of the neural interest map, either the SRN or the Kohonen feature map does well. Both have shown to be pretty good in such clustering problems but both run out of addressing space. An advantage of the Kohonen feature map might be a faster and more stable convergence although they have not been used for large (10.000 neurons) feature maps. Recent simulations of SRN's in IR showed very long training times and unstable behavior for large data sets. Advanced training techniques may suppress these effects for a while, but they remain deadly in the long run [Elman, 1991a-b].

If one wants to learn a specific mapping or function approximation, then back-propagation seems to be the best choice. However, one has to realize that most IR problems are clustering problems and not mapping problems, making BP a second choice.

6.8 Future Research

6.8.1 Hierarchical Feature Maps

Hierarchical feature maps are constructions of single feature maps that are organized in hierarchical structures. Main issues are the interconnection schemes and the training rules. Boundary effects are known to be very important in the organization process in a feature map. Large feature maps have relatively less boundaries and shall therefore entangle earlier. By combining feature maps in hierarchical sets of smaller feature maps, one can build large feature maps with many borders needed for the proper self-organization.

Many researchers have used the concept of hierarchical feature maps to represent relations between concepts [Miikulainen, 1990a,b]. However, the main problem is the connections between the feature maps; how does one define them and how are they incorporated in the training process? More on hierarchical maps can be found in [Kangas et al., 1990], [Kangas, 1990], [Koikkalainen et al., 1990], [Samarabunda et al., 1990], [Stotzka et al., 1990], [Ichiki et al., 1991], [Kohonen, 1991].

On the other hand, if the maps are derived (manually or automatically) they provide a great tool for the integration of knowledge structures in information retrieval. These feature maps might be on their own or they can be combined with the automatic derivation of synonym groups.More on the classic ideas in hierarchical document organization can be found in [Jardine et al., 1971], [Willett, 1979, 1984, 1988]. Currently, clustering of documents is considered expensive and irrelevant by most information retrieval experts.

6.8.2 Growing Network Structures

The most promising and most important future research in neural information retrieval is the evaluation of growing network structures. By automatically deriving the best (clustered) structure for a specific probability distribution., the effects of the neighborhoods in the neural filter as well as the problems with the cluster boundaries in the neural interest map may be solved in an elegant way.

The main property of this algorithm is that the feature map is always in order. Therefore, the map can never entangle and converge to improper values or form improper clusters [Fritzke, 1991a-b, 1992a-b], [Fritzke et al., 1991], [Martinetz et al., 1991].

6.8.3 Precision and Recall

Instead of comparing precision and recall values for different corpora (as it is done here), it would be interesting to compare results of a statistical analysis of the Pravda corpus to the neural values.

Chapter 5

Neural Data-Oriented Parsing

"Good copies are better than bad inventions"
- Old Japanese saying

Abstract

In a Data Oriented Parsing (DOP) system sentences are parsed on the basis of language examples from a large analyzed corpus. Parsing consists of deriving the most probable structure from fragments that already exist in the corpus. The DOP paradigm uses statistical properties of a structured corpus to assign structures and to resolve linguistic ambiguities. This in contrast to traditional rule-based systems. The main difference with other systems in corpus-linguistics is the incorporation of statistical information as well as structural and even semantic features in the training phase. This enables the model to rank ambiguous parses in order of preference on the basis of the features found in the corpus and in the input data.

This work describes an implementation of such a system that uses neural nets. The derivation of the "conditional probabilities" and the storage of the corpus is done by means of a Kohonen feature map. The actual data-oriented parsing is performed by a regular sequential algorithm. The model can process incomplete sentences, ungrammatical sentences and completely new sentences which contain a certain number of unknown words or structures. Implicit features of the Kohonen model turn the ranking of possible parses and the selection of the most probable one into a straightforward task.

1.0 Background

Recent surveys of linguistic research show a growing interest in statistically based linguistic theories. Some of these incorporate statistical ideas in a Chomskian framework [Smith, 1973]. Others use plain statistics without any notion of grammar rules to process language [Jelinek et al., 1989]. Without doubt, the reason for this change in research direction is the lack of good enough results in natural language processing obtained by the computational linguistic community. Chomsky's ideas were meant to describe linguistic structure. They were never really intended to process language or to derive a parse tree automatically. The inherent ambiguity of natural language is just too large to use grammar-rule systems efficiently [Lieberman, 1991].

It should be noticed however, that in the 50s many language processing methods were statistical in nature. The theory put forward by [Chomsky, 1957] opposed itself strongly against such statistical methods by providing solutions for problems that could not be solved with statistical methods. So, why should statistical theories work now, if they didn't work before? First, there are new mathematical methods such as re-estimation methods for Hidden Markov Models and Stochastic Context-Free Grammars. Secondly, computer systems have become so powerful, that it is possible to simulate statistical models on a large enough scale. Finally, large corpora to derive statistical information are widely available. Where [Shannon, 1951] calculated the entropy of the English language on a corpus of a few hundred characters, [Brown et al., 1990a] base their calculations on a 380 million words corpus.

Natural language has structure. This fact is accepted and honored by all computational linguists. However, it is hard to teach this notion to a computer system automatically or to incorporate it in the processing and still maintain reasonable performance. On the other hand, one needs structure to process language adequately and to resolve ambiguities. The main problem in statistically based linguistic processing is the incorporation of structural information.

Here, a corpus based parsing method is presented. By using a large structured corpus, sentences can be parsed properly. By doing so, the advantages of fast and trainable statistics are combined with powerful structural analysis of natural language. For the implementation of the corpus, a Kohonen feature map is used. The actual parsing is done by a regular sequential algorithm machine. The feature map is mainly used as an associative memory or an adaptive hashing function on all possible sentence analyses.

Parts of this work have been published in [Scholtes 1992e-h] and [Scholtes et al., 1992a-b].

2.0 Introduction

2.1 Data Oriented Parsing

The Data Oriented Parsing (DOP) paradigm was introduced by [Scha, 1990]. This paper states that "most language processing systems are linguistically motivated competence models of natural language". Instead, Scha proposes a "more performance oriented model of language processing that takes into account the statistical properties of real language use". This model is supposed to find the most probable analysis of a sentence in terms of fragments that are already in a corpus. Thus, the corpus is not only tagged with syntactic categories, but it also contains structural and even semantic features for every sentence fragment.

This in contrast to ,for example, [Garside et al., 1987], [Hanson et al., 1987] and [Church, 1988] who use corpora without structure. The main advantage of the presence of structure is that it enables the parser to concatenate different fragments in a better motivated, less ambiguous way than it would be able to do without such information. Moreover, we feel that a flat corpus cannot be a component of a plausible conceptual model of human language experience. We remember an utterance together with its context and its interpretation [Bod, 1992].

In short, a Data Oriented Parsing system uses a large analyzed corpus of natural language. A sentence is parsed by matching it against the corpus. First, all possible partitions of the sentence are generated. These partitions contain adjacent sentence fragments. Next, for each partition, the structures of the individual segments and their probabilities are determined by matching the fragments against the segments in the corpus.

By applying this process recursively, the possible analyses for the whole sentence, and their probabilities, can be determined. The most probable parse is then considered to be the preferred one.

DOP is a philosophy, not an implementation. Therefore, different realizations are possible. The most obvious one would explicitly compute conditional probabilities for extensions of syntactic trees, or would estimate these probabilities using the Monte Carlo method. Here, the emphasis is on the implementation of a data-oriented parsing system with a neural network.

2.2 Kohonen Feature Maps

The Kohonen formalism is a competitive learning algorithm [Kohonen, 1984, 1990b]. A two-dimensional map is constructed in a rectangular or hexagonal structure from individual neurons. Each neuron has a number of input sensors with an input activation and an input weight. All neurons have the same number of input sensors. The learning rule acts in the following way. First, copy the activation values of an input element into all input activation sensors of all neurons. Next, determine the best match by finding the neuron with the minimum distance between input and weight values. Then, adapt the weights of the neurons within a certain region of this minimum, so they recognize the current input vector better in the future. After numerous cycles, a topological map is formed, holding related elements in neighboring regions.

Previous work showed that Kohonen feature maps organize on frequency as well as on similarity, rising the model above the level of simple frequency based probability models. The values that the weights in the Kohonen map represent are therefore better than clean probabilities. By implementing a complex self-organizing process, simple concept formations can be derived automatically [Ritter et al., 1989b, 1990], [Scholtes, 1991b-m, 1992a-d]. Moreover, the property that all sentence-fragments can be trained in linear time and space with respect to the fragment size, makes Kohonen feature maps well suited for an efficient implementation of the corpus.

3.0 The Model

The model consists of a single layer Kohonen feature map. The sensors of the individual neurons are grouped in small sets. Each set represents one element of a shifting window (see figure 1, "The training model of the neural data oriented parsing corpus"). During training, a sentence is selected from the corpus at random. Next, a window of size k is shifted over the sentence. Depending on the number of training cycles, an individual sentence in the corpus is selected a number of times for training. The corpus is organized in such a way, that each word is tagged with its syntactical category as well as with the labels of its dominating nodes. Each time a sentence is selected for training, all these features are taught to the net.

The model incorporates context in two different ways. First there is the shifting window that provides a simple word sequence context. The second contextual influence is more complicated. During training, the model is exposed to random sentences from the corpus. Then, sentence fragments shall be organized so that related fragments are stored in neighboring regions. By making the model a little smaller than the required corpus size, less frequent fragments as well as fragments that occur in less frequent contexts are absorbed by more frequent and more common contexts. So, the value that is fired by the neurons in the retrieval phase, shall represent frequency as well as context.

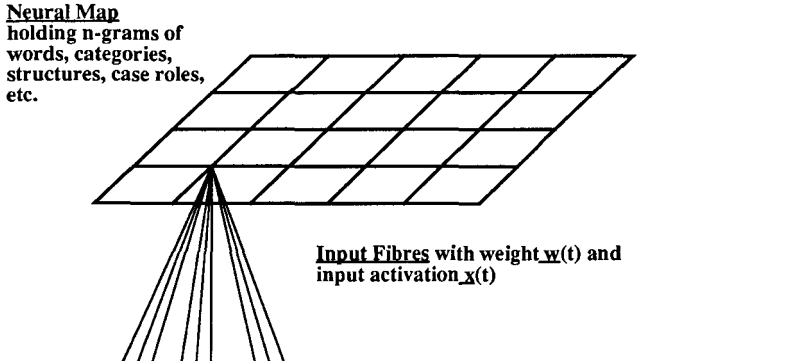


FIGURE 1. The training model of the neural data oriented parsing corpus

By using a special coding mechanism, structures can be represented by a representation as shown below (see figure 2, "The mapping from structure to a concatenation of vectors").[1] Other structural descriptors as well as other coding schemes can be incorporated easily in this framework. The only restriction is that the code should support the possibility for the Kohonen feature map to organize the data in such a way that related items are in neighboring regions. In other words, if two fragments have a similar structure, the coding should also be similar in such a manner that the Euclidean distance of the two encodings is small. In the coding scheme used here, the structure S is mapped to the matrix M, where M is the concatenation of V_1, up to V_5:

$$M = V_1 \oplus V_2 \oplus V_3 \oplus V_4 \oplus V_5 \qquad \text{(EQ 1)}$$

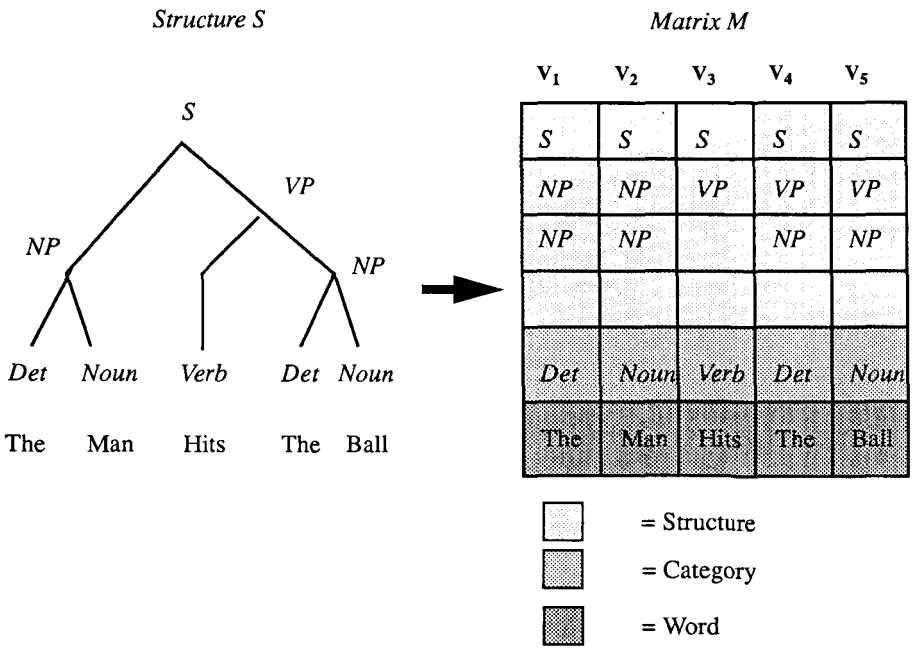


FIGURE 2. The mapping from structure to a concatenation of vectors

1. The translation from a hierarchical structure to a vector, as used here, is a very straightforward and simple one. There is much research going on towards these kind of mappings. Some of them are already explained in Chapter 2: "Neural Networks in Natural Language Processing and Information Retrieval". Here, it is not the intension to evaluate these kinds of mappings in depth. Therefore, a mapping has been selected that fits best to the Kohonen model. However, some other mapping techniques from hierarchical structures to vectors have also been evaluated. But, the results were not as good as in the case presented.

The model is trained by shifting a window of fixed size over sentences in the corpus. Inter-sentence relations are not incorporated; all sentences are treated as separate cases. Every sentence is tagged with information such as word categories, sentence structure and other (semantic) features. The more information the better. After training, the Kohonen feature map can be viewed as a number of stacked maps with one layer[1] for every window element. Every sensor has k layers, where k is the size of the shifting window. Every layer can be divided into three sub-layers: a word code, a category code and a structure code (see figure 3, "A sentence fragment as a column in the model").

After training, the model holds all possible unique fragments of the corpus (as long as the number of neurons is large enough). On every layer, comparable fragments will be organized into neighboring layers. So, if one divides the model in different slices, groups of nouns, verbs, NPs, etc. can be distinguished clearly in neighboring regions.

The actual parsing is done by using a conventional Von Neumann machine that accesses the Kohonen feature map as an associative memory for its corpus. The corpus is clearly separated from the parsing module. By doing so, different techniques for the corpus as well as the parsing can be tested easily (see section 5.2 "The Parsing Algorithm" for details).

1. Although the word layer is used, it is not the intention to suggest the use of a multi-layer Kohonen model. Nevertheless, every element of the shifting window can be seen as part of a separate Kohonen map. Thus the elements of a window are organized in different layers. Every (window-size) sentence fragment is stored as a column within these layers.

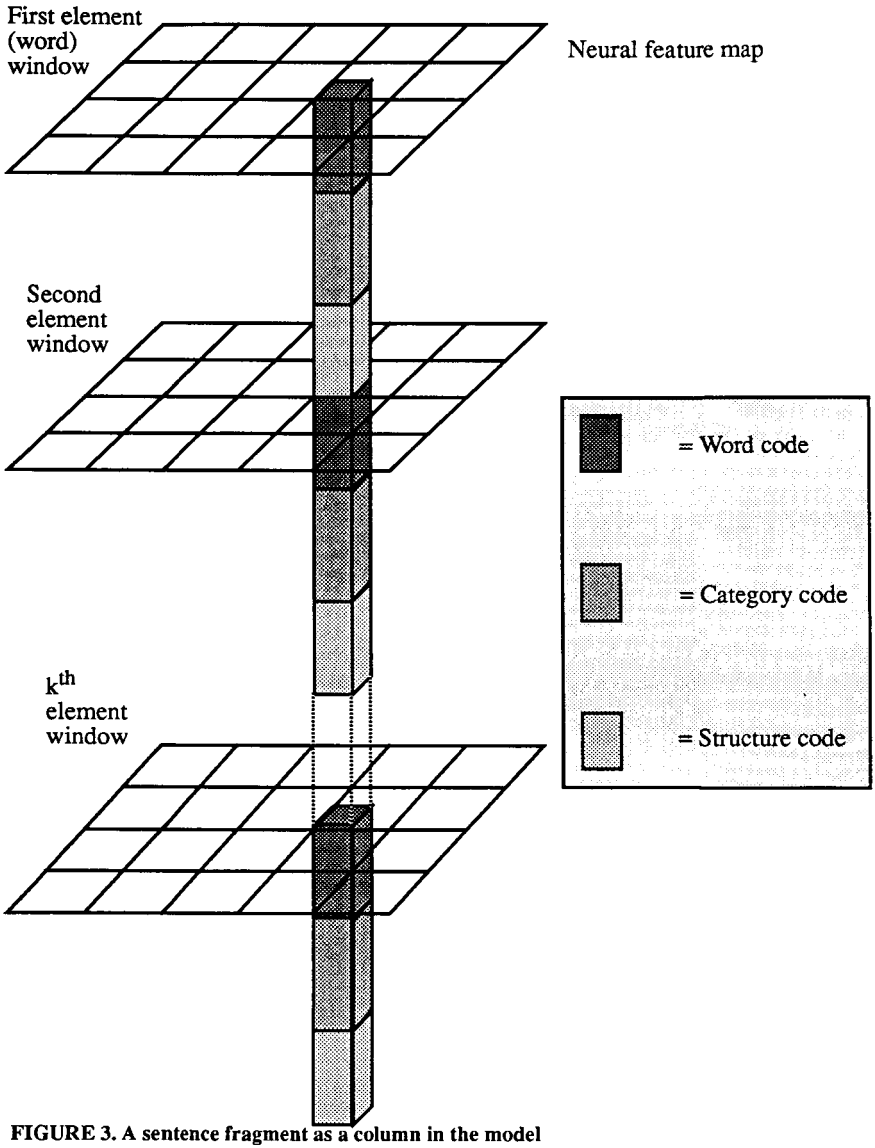


FIGURE 3. A sentence fragment as a column in the model

4.0 Corpus Representation

In this section, the technical details of the DOP implementations shall be discussed in detail. First, a representation scheme for a context free grammar shall be given. Next, two extensions of the model are discussed, making it a linguistically more plausible one. In the first extension, besides syntactic features, semantic properties are added to the analyses of sentences in the corpus in the form of case role assignments. In the second extension, non-terminals are included in the training set, enabling the model to substitute subtrees of different word lengths.

Throughout the entire corpus, all morphology is eliminated. The sentence "The man hits the ball" is represented as "The man hit the ball".

4.1 Type 1: Straightforward Encoding

Assume the corpus is annotated with consistent trees in a common, Chomskyan format (see Figure 2, "The mapping from structure to a concatenation of vectors" on page 206). For the purpose of this project, these tree-structures were initially translated into the flat structure as shown in Table 1, "layout corpus 1". This was the most straightforward method possible. It adopts the Kohonen formalism very well by encoding similar objects with similar codes.

TABLE 1. layout corpus 1

Word	Category	Structure	Structure	Structure	Structure	Structure	Structure
THE	DET	E	E	E	NP	NP	S
MAN	NOUN	E	E	E	NP	NP	S
AND	CONJ	E	E	E	E	NP	S
THE	DET	E	E	E	NP	NP	S
WOMAN	NOUN	E	E	E	NP	NP	S
RUN	VERB	E	E	E	E	VP	S
INTO	PREP	E	E	E	PP	VP	S
THE	DET	E	E	NP	PP	VP	S
GARDEN	NOUN	E	E	NP	PP	VP	S

However, this format has three main disadvantages:

- It can only cover trees of a maximum tree-depth.
- In the case of categories governing on two nodes of the same type (like bitransitive verbs), these arguments will be represented at the same level, thus resulting in an ambiguous and even non-sensical translation of an unambiguous tree.

- Only substitutions of the same length can be processed by the model. Whenever one would e.g. substitute a large NP such as "the old man and the woman" by "the man", the model would not support this. This limitation is mainly due to the fact that non-terminals (such as NP, VP, and PP) are not included in the terminal level during training.

4.2 Type 2: Case Role Assignment

Type 2 is an extension of type 1 with semantical features to enable the processing of bitransitive verbs.

A method well known from semantics is case role assignment. Every syntactical group is assigned an semantical role in the sentence. The roles may be described in different ways. The cases used here are given in Table 2, "Cases used for case role assignment tagging of the type 2 corpus".

TABLE 2. Cases used for case role assignment tagging of the type 2 corpus

Tag	Semantical name	Purpose or meaning
SEM	-	No specific semantical meaning
AGENT	Agent	The object performing the action
LOCAT	Location	Indication of a location
INSTR	Instrument	The instrument with which the action is performed
THEME	Theme	The object that is the theme of the action
PAT	Patient	The object that is modified by the action
GOAL	Goal	Indication of a direction or destination
MAN	Manner	Modifier on a verb or verb phrase, like "better", "faster", etc.

TABLE 3. Type 2: Type 1 tagged with semantical features

Word	Category	St	CR	St	CR	St	CR	St	CR	St	CR	ST	CR
THE	DET	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM	S	SEM
MAN	N	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM	S	SEM
BEAT	V	E	SEM	E	SEM	E	SEM	VP	SEM	S	SEM	S	SEM
THE	DET	E	SEM	E	SEM	NP	PAT	VP	SEM	S	SEM	S	SEM
CAT	N	E	SEM	E	SEM	NP	PAT	VP	SEM	S	SEM	S	SEM
IN	PREP	E	SEM	E	SEM	E	SEM	E	SEM	PP	LOCAT	S	SEM
THE	DET	E	SEM	E	SEM	E	SEM	NP	SEM	PP	LOCAT	S	SEM
GARDEN	N	E	SEM	E	SEM	E	SEM	NP	SEM	PP	LOCAT	S	SEM
WITH	CONJ	E	SEM	E	SEM	E	SEM	NP	SEM	PP	INSTR	S	SEM
THE	DET	E	SEM	E	SEM	E	SEM	NP	SEM	PP	INSTR	S	SEM
BOOK	N	E	SEM	E	SEM	E	SEM	NP	SEM	PP	INSTR	S	SEM

By tagging every syntactical element, the coding scheme as presented in Table 3, "Type 2: Type 1 tagged with semantical features" is obtained (ST holds the structural information on the element, CR the case role assignment). The advantage of the tagging can be seen in the sentence: "the woman gives the child the book". In this sentence, two different NP's are occurring on the same level. Before, the model could not separate the end from the first NP from the beginning of the second. As a result, "the child the book" is considered a legal NP, which is of course wrong. By using the semantical tags, the model can separate these two NP's from each other. See Table 4, "Type 2 representation of bitransitive verbs" for the details of the implementation.

TABLE 4. Type 2 representation of bitransitive verbs

Word	Category	ST	CR	ST	CR	ST	CR	ST	CR	ST	CR	ST	CR
THE	DET	I	SEM	I	SEM	I	SEM	I	SEM	NP	AGENT	S	SEM
WOMAN	N	E	SEM	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM
GIVE	V	E	SEM	E	SEM	E	SEM	E	SEM	VP	SEM	S	SEM
THE	DET	E	SEM	E	SEM	E	SEM	NP	GOAL	VP	SEM	S	SEM
CHILD	N	E	SEM	E	SEM	E	SEM	NP	GOAL	VP	SEM	S	SEM
THE	DET	E	SEM	E	SEM	E	SEM	NP	THEME	VP	SEM	S	SEM
BOOK	N	E	SEM	E	SEM	E	SEM	NP	THEME	VP	SEM	S	SEM

4.3 Type 3: Non-Terminals

So, the type 2 corpus is able to process sentences with ambiguous verb arguments. However, the problem of unequal length substring substitution has not been solved yet. In order to do this, the model should also be trained on the terminal level with non-terminals such as NP, VP, etc. By doing so, the analysis that is returned by the Kohonen feature map might contain non-terminals, which could then be substituted by unequal length substrings of terminals by an external shell.

TABLE 5. Abstractions of one sentence with all the non-terminals

Word	Category	St	CR	ST	CR	ST	CR	ST	CR	ST	CR	ST	CR
THE	DET	E	SEM	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM
MAN	N	E	SEM	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM
SEARCH	V	E	SEM	E	SEM	E	SEM	E	SEM	VP	SEM	S	SEM
THE	DET	E	SEM	E	SEM	E	SEM	NP	THEME	VP	SEM	S	SEM
DOG	N	E	SEM	E	SEM	E	SEM	NP	THEME	VP	SEM	S	SEM

TABLE 5. Abstractions of one sentence with all the non-terminals

NP	NP	E	SEM	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM
SEARCH	V	E	SEM	E	SEM	E	SEM	E	SEM	VP	SEM	S	SEM
THE	DET	E	SEM	E	SEM	E	SEM	NP	THEME	VP	SEM	S	SEM
DOG	N	E	SEM	E	SEM	E	SEM	NP	THEME	VP	SEM	S	SEM

THE	DET	E	SEM	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM
MAN	N	E	SEM	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM
SEARCH	V	E	SEM	E	SEM	E	SEM	E	SEM	VP	SEM	S	SEM
NP	NP	E	SEM	E	SEM	E	SEM	NP	THEME	VP	SEM	S	SEM

THE	DET	E	SEM	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM
MAN	N	E	SEM	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM
VP	VP	E	SEM	E	SEM	E	SEM	E	SEM	VP	SEM	S	SEM

NP	NP	E	SEM	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM
SEARCH	V	E	SEM	E	SEM	E	SEM	E	SEM	VP	SEM	S	SEM
NP	NP	E	SEM	E	SEM	E	SEM	NP	THEME	VP	SEM	S	SEM

NP	NP	E	SEM	E	SEM	E	SEM	E	SEM	NP	AGENT	S	SEM
VP	VP	E	SEM	E	SEM	E	SEM	E	SEM	VP	SEM	S	SEM

A limited number of sentences from the entire corpus has been abstracted towards a set of all possible combinations between terminals and non-terminals. In Table 5, "Abstractions of one sentence with all the non-terminals" an example can be found for the sentence "the man search the dog". In this representation scheme, the model will primarily be used as an associative memory from which possible analyses are obtained. The assignment of terminal-strings to the non-terminals should finally be done by a method as introduced by [Bod, 1992], where a Monte Carlo method is used to select proper terminals from the corpus.

4.4 Other Extensions of the Corpus

Other possible extensions to the corpus might be the addition of other semantical, discourse or even visual information. As a result, the DOP corpus could be regarded as a true collection of different knowledge types which all participate concurrently during parsing. So, the neural implementation of the DOP corpus implements what is mentioned in the first chapters as a uniform knowledge representation method, regardless of the type and character of the knowledge or facts used.

4.5 Types Evaluated

All corpus representation schemes that are used in the simulations presented here, use the basic tree encoding principles of the type 1 representation. However, there has been much research towards the representation of, for example, binary trees in vectors.

Some of these representation methods have also been used (see for example [Scholtes et al., 1992b]. But, it appeared that these other encoding schemes did not follow the Kohonen principles as good as the coding that is used here does. Therefore, the results obtained in those simulations are not mentioned here.

The three types which are evaluated in the simulations can be found in Table 6, "Different types of corpus representations". The training and parsing algorithms belonging to these input types as well as the results shall be discussed in the following sections.

TABLE 6. Different types of corpus representations

Corpus Number	Description
1	Straightforward encoding
2	Corpus 1 tagged with semantical features
3	Corpus 2 extended with the training of non-terminals

5.0 Training and Parsing Algorithm

For all 3 different corpus representation schemes, a similar algorithm of training and retrieval was used. The only difference was in the type of information encoded in the vectors taught to the neural network.

5.1 The Training Algorithm

In all 3 training methods, a window of size k (in most of the simulations, $k = 4$) was shifted over a randomly selected sentence S of the corpus. The size of the window k that was used in these simulations was 4. This value was determined by experimental means. The vector that was presented to the training algorithm consisted of a number of different concatenations. First there was the concatenation of the vector representation of the different tags per word φ_i. For the different types, these concatenations can be found in Table 7, "Elements part of the training vector per corpus representation scheme".

TABLE 7. Elements part of the training vector per corpus representation scheme

Type	Concatenation φ_i
1	$word \oplus category \oplus structure$
2	$word \oplus category \oplus structure \oplus case-role$
3	$(word \vee non-terminal) \oplus category \oplus structure \oplus case-role$

In addition, the result of the k elements that are shifted over the sentence are also concatenated into one large vector. So, if a sentence consists of n words, and $n>k$, then n-k windows are obtained that are all presented once to the feature map. See Table 8, "Elements of one sentence presented to the network" for an example.

TABLE 8. Elements of one sentence presented to the network

Window Number	Contents
1	$\varphi_1 \oplus \varphi_2 \oplus \ldots \oplus \varphi_k$
2	$\varphi_2 \oplus \varphi_3 \oplus \ldots \oplus \varphi_{k+1}$
...	
...	
n-k	$\varphi_{n-k} \oplus \varphi_{n-k+1} \oplus \ldots \oplus \varphi_n$

where φ_i represents the coding of the i^{th} element of sentence S.

Then, through some lookup tables, the vector codes for these symbolic elements are used to construct a large numerical vector that is used by the training algorithm. All elements of the training set are assigned randomly to specific codes in this lookup table. The codes themselves are spread homogeneously through the feature space, to speed up the training process. Once again, the training rule used is the Kohonen rule. Convergence parameters as proposed by [Ritter et al., 1989a] fine-tune the Kohonen rule.

5.1.1 The Kohonen Training Rule

For the sake of completeness, the Kohonen training rule is repeated here. Suppose that the complete encoded concatenation, representing a substring of the sentence presented at time step t, is represented by $\underline{x}(t)$. In addition, let $\underline{w}_i(t)$ represent the weights of neuron i at feature map M. Then, the algorithm goes as follows: first determine the minimum of the feature M: neuron v. This neuron is the Best Matching Unit (BMU) between the input values and its weight values:

$$\text{v:} \qquad \forall k \left(\left\| \underline{w}_v(t) - \underline{x}(t) \right\| \leq \left\| \underline{w}_k(t) - \underline{x}(t) \right\| \right) \tag{EQ 2}$$

for k is a neuron on the feature map.

Next, update all weights in the map according to the actual Kohonen training rule:

$$\underline{w}_k(t+1) = \underline{w}_k(t) + \varepsilon(t) \cdot \Phi_{kv} \cdot \left(\underline{x}(t) - \underline{w}_k(t) \right) \tag{EQ 3}$$

where

$$\Phi_{rs} = e^{-\| k - v \| \cdot 2\sigma^2(t)} \tag{EQ 4}$$

$$\varepsilon(t) = \varepsilon_{max} \cdot \left(\frac{\varepsilon_{min}}{\varepsilon_{max}} \right)^{\frac{t}{t_{max}}} \tag{EQ 5}$$

$$\sigma(t) = \sigma_{max} \cdot \left(\frac{\sigma_{min}}{\sigma_{max}} \right)^{\frac{t}{t_{max}}} \tag{EQ 6}$$

where

$\omega \in [0,1]$ Memory Rate,

$\varepsilon_{max} \in [0,1]$ Initial Learning Rate,

$\varepsilon_{min} \in [0,1]$ Final Learning Rate,

$\sigma_{max} \in [0,1]$ Initial Region Size

$\sigma_{min} \in [0,1]$ Final Region Size, and

$\|k - v\|$ Physical distance between neuron k and neuron v

In advance, a predetermined number of training cycles is set. Then, one by one, random sentences are selected from the corpus and present to the feature map. After training, all equal length substrings with the same features shall be located in neighboring areas on the feature map. Thus, this organization that is implemented shall be based on features such as word, category, governing categories, case role, or on whatever combination of these four.

5.2 The Parsing Algorithm

In this section, the parsing process shall be discussed in detail. Before this can be done properly, a number of definitions need to be given to properly understand the meaning of the terms used (see Table 9, "Definition of the terms used").

TABLE 9. Definition of the terms used

Name	Definition
Sentence	The total input string to be analyzed, viewed as a sequence of words. Supposed to be one linguistic unit.
Fragment	Sequence of words that constitutes a subsequence of a sentence.
Partition	Sequence of fragments that precisely covers a sentence. For sentence ABC, all possible partitions are: <ABC>, <AB,C>, <A,BC>, <A, B, C>.
Segment	A fragment may be too small or too large for feed-forward into the neural network. Therefore, it needs to be converted to a number of segments that can straightforwardly be compared with the neural network. If the network is trained by using windows of size *3*, then the fragment A of one element must be converted to the 3 possible segments: <A,-,->, <-,A,-> and <-,-,A>. If the string would be ABCD, than it would have been converted to the segments <ABC>, and <BCD>. The result of the analyses of the fragment would then be some function of the results of the segments.
Parse	The combined result of the analyses of all segments into fragments per partition
Post-processing	A second feed-forward of the analyses per partition to avoid the construction of nonsense structures and to determine the ranking of the analyses with respect to each other.
Ranking	Ranking of the partitions with respect to the error of the analyses in the postprocessing phase.

One should be aware that the neural network is mainly used as an efficient implementation of the corpus. The actual parsing is done by an external shell. In fact, a regular sequential algorithm machine generates the partitions, fragments and segments; then these are fed-

forward in the neural network. The results are combined by the same sequential algorithm into a sentence analysis. In the following sections, the partition algorithm shall be discussed in detail. The number in the text points to the step in Table 10, "Global overview of the parsing algorithm".

First, all sentence partitions are generated by an external shell. In total 2^{n-1} partitions are generated. The derivation of the most probable parse can be obtained by determining the partition with the lowest complexity [Step 1].

The partition with the lowest complexity is the one with the lowest error as a function of its fragments. So, one needs to determine the set of unique sentence fragments, generated by the partitions. In total, $\frac{1}{2}n^2 + \frac{1}{2}n$ different fragments can be distinguished. These fragments can be evaluated one by one, after which the evaluation per partition can be calculated [step 2].

Whenever a fragment needs to be evaluated, a technical problem might occur. Suppose the feature map is trained with a *k*-size window. then the corpus only contains fragments of size *k*. Fragments of sizes smaller or larger than *k* cannot be evaluated without adapting them to the size of the fragments present in the corpus. So, fragments larger than size *k* are shifted along the corpus. Fragments smaller than size *k* are extended to length *k* and are then evaluated. As an example, if the network is trained by using windows of size *3*, then the fragment A of one element must be converted to the 3 possible segments: <A,-,->, <-,A,-> and <-,-,A>. If the string would be <ABCD>, than it would have been converted to the segments <ABC>, and <BCD>. The evaluation of a fragment is then a function of the evaluations of the segments [Step 3].

All sentence fragments are then evaluated by matching the corresponding segments to the neural net. For each segment, the word part of the vector is matched. It will fit best on a particular neuron. The parse belonging to the sentence segment is then the parse represented by the category, structure, and semantic part of the neuron weight vector.

However, there will always be a slight matching error: $\|w - x\|$. The lower this error, the better the neuron represents the segment. The segment word, category, structure and semantics are returned together with the error are then combined into an overall evaluation of the fragment. By eliminating all parses for fragments of which one of the segments has a high error, most nonsense structures are eliminated.

Next, the segments need to be combined into one fragment. This can be done by several means (see figure 4, "A possible transformation from segments to fragments"). A possibility is to fill each slot with the most frequent occurrence for this slot in all segments. Another possibility is to fill each slot with the best occurrence in all the segments (the one with the lowest error) [Step 4].

```
Partition Word Representation:
     THE      MAN   SEARCH      THE
              MAN   SEARCH      THE      DOG
                       SEE      A        CAT       IN
                                THE      CAT       IN       THE
                                         CAT       IN       THE     GARDEN
Best Words:
     THE      MAN   SEARCH      THE      CAT       IN       THE     GARDEN

Partition Category Representation:
     DET        N         V     DET
                N         V     DET        N
                          V     DET        N      PREP
                                DET        N      PREP      DET
                                           N      PREP      DET        N
 Best Categories:
     DET        N         V     DET        N      PREP      DET        N
```

FIGURE 4. A possible transformation from segments to fragments

As easy as the segments can be combined into a fragment, the fragments can be combined into partitions. Here too, partitions which contain high errors or eliminated fragments are not generating valid parses, and are therefore eliminated [Step 5].

A post-processing phase is needed to avoid the generation of nonsense parses due to the fact that different neurons which fire on the subsequent sub-partitions can represent non-consistent structures. To avoid the construction of these nonsense structures, the sentences are then once again fed-forward into the network in the same manner as they were trained. Nonsense parses generate unacceptably high errors and are thus eliminated by the post processing module.

In the case of type 4 corpus representation, the situation is even more complex. The neural network can return terminals as well as non-terminals in the final sentence analysis. As a result, the sentences that contain non-terminals should be post-processed by another external shell, which uses some kind of (random) algorithm to exchange the non-terminals by possible substrings that can be derived from the neural network. In particular this extension of the algorithm, makes it into a flexible parsing model that can also derive sentence parses for sentences or sentence structures it has never seen before [Step 6].

The total error of the matching of these structures is then analyzed by the external shell, which consequently ranks the sentences in the order of minimum error, which is the same

as the order of the most preferable parse given the context of all possible features in which the sentence occurs [Step 7].

TABLE 10. Global overview of the parsing algorithm

Step	Description
1	Generate all possible adjacent sentence partitions. The total number of partitions is given by the formula: 2^{n-1} where n is the sentence length in words.
2	Determine the set of possible fragments of which all partitions are constructed. The total number of fragments is given by the formula: $\frac{1}{2}n^2+\frac{1}{2}n$, where n is the sentence length in words.
3	For all sentence fragments, generate the segments needed to compare to the neural network. The number of segments per fragment depends on the size of the window during training.
4	Feed-forward all sentence fragments by matching the corresponding segments with the neural nets. Combine the results of the segment and determine the error per fragment.
5	For all partitions, combine the results of the fragments into one sentence analysis if the error of the (combination of the) fragments is smaller than a certain threshold.
6	Feed-forward all partition results for which the total error of all the fragments is smaller than a certain threshold. If the total result of this post-processing process is smaller than a certain threshold, keep the analysis, otherwise, throw it away.
7	Rank all the accepted analyses in order of preference by taking into account the feed-forward as well as the combined fragment error

Especially the determination of the error from the segments into the fragments into the partitions can be implemented in many different manners. In addition, the determination of the thresholds used in the elimination process of nonsense analyses is a process that has to be done by trial and error, although it is not as difficult as one should expect.

6.0 Example of a Sentence Parse

In this section, the parse of the sentence: "The man search the dog" shall be given in full detail. This in order to increase the readability of this chapter.

First, the sentence is split into a set of sentence partitions.

TABLE 11. Sentence Partitions

Nr..	Partitions				
0	THE MAN SEARCH THE DOG				
1	THE MAN SEARCH THE	DOG			
2	THE MAN SEARCH	THE DOG			
3	THE MAN SEARCH	THE	DOG		
4	THE MAN	SEARCH THE DOG			
5	THE MAN	SEARCH THE	DOG		
6	THE MAN	SEARCH	THE DOG		
7	THE MAN	SEARCH	THE	DOG	
8	THE	MAN SEARCH THE DOG			
9	THE	MAN SEARCH THE	DOG		
10	THE	MAN SEARCH	THE DOG		
11	THE	MAN SEARCH	THE	DOG	
12	THE	MAN	SEARCH THE DOG		
13	THE	MAN	SEARCH THE	DOG	
14	THE	MAN	SEARCH	THE DOG	
15	THE	MAN	SEARCH	THE	DOG

Next, a number of unique sentence fragments is determined. Each of these sentence fragments is converted to a set of sentence segments, which are fed forward into the neural net. All these sentence segments have a best matching unit (BMU)with a corresponding error:

TABLE 12. Evaluation of sentence fragments

Nr.	Fragment	BMU	Error
1	THE		
		(0, 8)	0.0000
		(0, 0)	0.0000
		(0,16)	0.0000
		(0,12)	0.0000

TABLE 12. Evaluation of sentence fragments

Nr.	Fragment	BMU	Error
2	THE MAN		
		(0,12)	0.0000
		(0, 0)	0.2500
		(4,14)	0.0000
3	THE MAN SEARCH		
		(4, 4)	0.0000
		(0, 0)	0.2500
4	THE MAN SEARCH THE		
		(4, 8)	0.0000
5	THE MAN SEARCH THE DOG		
		(4, 8)	0.0000
		(2,10)	0.0000
6	MAN		
		(2, 0)	0.0000
		(0,12)	0.0000
		(0, 8)	0.0000
		(4,14)	0.0000
7	MAN SEARCH		
		(2,10)	0.0000
		(4, 4)	0.0000
		(0, 0)	0.2500
8	MAN SEARCH THE		
		(2,10)	0.0000
		(4, 8)	0.0000
9	MAN SEARCH THE DOG		
		(2,10)	0.0000
10	SEARCH		
		(3, 9)	0.2500
		(0,16)	0.0000
		(2, 6)	0.0000
		(0, 0)	0.0000
11	SEARCH THE		
		(1, 4)	0.3417
		(0,16)	0.0000
		(4, 6)	0.0000
12	SEARCH THE DOG		
		(3, 8)	0.5925
		(0,16)	0.0000
13	THE		

TABLE 12. Evaluation of sentence fragments

Nr.	Fragment	BMU	Error
		(0, 8)	0.0000
		(0, 0)	0.0000
		(0,16)	0.0000
		(0,12)	0.0000
14	THE DOG		
		(9, 8)	0.5329
		(3, 8)	0.5195
		(0,16)	0.0000
15	DOG		
		(6, 6)	0.5000
		(3,11)	0.3536
		(5, 9)	0.2705
		(0,16)	0.0000

For all these sentence fragments/segments the corresponding word, syntactic category and syntactic structure are determined. As an example, the fragment "The man search the dog" generates two segments: "The man search the" and "man search the dog" which are both fed forward in the net (fragment number 5 from the previous table). Then, a number of best matching words, syntactic categories and structures can be determined taking the maximum number of best fitting elements for each fragment.

TABLE 13. Fragment word representation

Fragment Word Representation:					
	THE	MAN	SEARCH	THE	
		MAN	SEARCH	THE	DOG
Best words:					
	THE	MAN	SEARCH	THE	DOG

TABLE 14. Fragment category representation

Fragment Category Representation:					
	DET	N	V	DET	
		N	V	DET	N
Best categories:					
	DET	N	V	DET	N

TABLE 15. Fragment structure representation

Fragment Structure Representation:					
	S	S	S	S	
	NP	NP	VP	VP	
	E	E	E	NP	
	E	E	E	E	
	E	E	E	E	
	E	E	E	E	
		S	S	S	S
		NP	VP	VP	VP
		E	E	NP	NP
		E	E	E	E
		E	E	E	E
		E	E	E	E
Best structures:					
	S	S	S	S	S
	NP	NP	VP	VP	VP
	E	E	E	NP	NP
	E	E	E	E	E
	E	E	E	E	E
	E	E	E	E	E

This process is repeated for all fragments, which are then combined in a similar manner to partitions. Below, an example is given for partition number 1 from Table 11, "Sentence Partitions" is given. Here too, the maximum number of best fitting elements per fragment

TABLE 16. Partition word representation

Partition Word Representation:				
THE	MAN	SEARCH	THE	
	MAN	SEARCH	THE	DOG
Best words:				
THE	MAN	SEARCH	THE	DOG

is selected to be represented in the final partition.

TABLE 17. Partition category representation

Partition Category Representation:				
DET	N	V	DET	
	N	V	DET	N
Best categories:				
DET	N	V	DET	N

TABLE 18. Partition structure representation

Partition Structure Representation:					
	S	S	S	S	
	NP	NP	VP	VP	
	E	E	E	NP	
	E	E	E	E	
	E	E	E	E	
	E	E	E	E	
		S	S	S	S
		NP	VP	VP	VP
		E	E	NP	NP
		E	E	E	E
		E	E	E	E
		E	E	E	E
Best structures:					
	S	S	S	S	S
	NP	NP	VP	VP	VP
	E	E	E	NP	NP
	E	E	E	E	E
	E	E	E	E	E
	E	E	E	E	E

Next, all partitions are ordered on basis of the total retrieval error, resulting in a preferred parse. Here, the 5 most preferred parses are given in order of preference:

TABLE 19. 5 most preferred parses

words	THE	MAN	SEARCH	THE	DOG
category	DET	N	V	DET	N
structure	S	S	S	S	S
	NP	NP	VP	VP	VP
	E	E	E	NP	NP
	E	E	E	E	E
	E	E	E	E	E
	E	E	E	E	E
words	THE	MAN	SEARCH	THE	DOG
category	DET	N	V	DET	N
structure	S	S	S	S	S
	NP	NP	VP	VP	VP
	E	E	E	NP	NP
	E	E	E	E	E
	E	E	E	E	E
	E	E	E	E	E
words	THE	MAN	SEARCH	NP	DOG
category	DET	N	V	NP	N
structure	S	S	S	S	S
	NP	NP	VP	VP	VP
	E	E	E	NP	NP
	E	E	E	E	E
	E	E	E	E	E
	E	E	E	E	E
words	THE	MAN	SEARCH	NP	PRETTY
category	DET	N	V	NP	ADJ
structure	S	S	S	S	S
	NP	NP	VP	VP	NP
	E	E	E	NP	NB
	E	E	E	E	E
	E	E	E	E	E
	E	E	E	E	E

TABLE 19. 5 most preferred parses

words	THE	MAN	AND	THE	DOG
category	DET	N	CONJ	DET	N
structure	S	S	S	S	S
	NP	NP	NP	NP	VP
	NP	NP	E	NP	NP
	E	E	E	E	E
	E	E	E	E	E
	E	E	E	E	E

7.0 Results

In this section, the results of the neural DOP simulations shall be discussed. The results are separated into five different subsections: one subsection with general results and then four additional subsections for the four different corpus representation schemes.

7.1 General Results of the Model

After training, fragments of size k (where k is the size of the window, $k = 4$ during the simulations presented here), are represented by the neurons in the Kohonen feature map. All levels represent similar objects in neighboring regions. In the vertical directions, fragments can be discovered, on the horizontal level, groups of concepts can be found. By incorporating the word, the syntactic category and the structure in which the word occurs, an organization is derived that can distinguish complex linguistic structures.

A first conclusion that can be drawn is that the corpus appeared to be of much higher quality when it was trained with the structures than without them.

The model was capable of ordering ambiguous parses in order of preference. Incorporating additional semantic features in the training and retrieval phase results in a context sensitive method for word-sense disambiguation. Especially this ranking ability and the related disambiguation power of the model seems to be a major advantage of the (neural) DOP paradigm.

7.2 Results Corpus 1: The Straightforward Implementation

The first corpus consisted of 69 different sentences that contained a total of 494 words. In total 36 different words, 13 unique syntactic word categories, and 13 structural (non-terminals) could be distinguished. The total number of training cycles varied between 150.000 and 300.000. No significant difference in performance between maps with these different training times could be observed.

After training, the words as well as the word categories and the structures formed clusters on the feature maps for each element of the window. The weights of this feature map were used to test several training algorithms, some of which are discussed in the rest of this section.

First, a normal sentence can be observed: "The man search the dog in the garden". In the next figure, an overview of the analyses is given in the format as generated by the computer simulation.

TABLE 20. Parse of the sentence: "The man search the dog in the garden"

THE	MAN	SEARCH	THE	DOG	IN	THE	GARDEN
DET	N	V	DET	N	PREP	DET	N
S	S	S	S	S	S	S	S
NP	NP	VP	VP	VP	PP	PP	PP
E	E	E	NP	NP	E	NP	NP
E	E	E	E	E	E	E	E
E	E	E	E	E	E	E	E
E	E	E	E	E	E	E	E

Data oriented parsing should process ungrammatical or misspelled sentences just as easily as correct sentences (up to a maximum number of wrong words). In the following example, the word "the" is incorrectly replaced by the word "teh" which is not known by the model. Nevertheless, the algorithm is capable of parsing the sentence correctly. It even generates only one sentence. No other reasonable analyses were available in the corpus.

TABLE 21. Parse of the misspelled sentence "Teh man and the woman search the dog"

THE	MAN	AND	THE	WOMAN	SEARCH	THE	DOG
DET	N	CONJ	DET	N	V	DET	N
S	S	S	S	S	S	S	S
NP	NP	NP	NP	NP	VP	VP	VP
NP	NP	E	NP	NP	E	NP	NP
E	E	E	E	E	E	E	E
E	E	E	E	E	E	E	E
E	E	E	E	E	E	E	E

In the following example, this property of the DOP model is tested for another word, which could generate more possibilities. Instead of "the man see the dog"[1], "The XXX see the dog" is entered. As can be seen, the model correctly generates a number of possibilities in order of probability:

"The man see the dog",

"The woman see the dog",

1. In the corpus, all morphology is eliminated. This means that instead of "The man sees...", the corpus is trained with "The man see...". Verbs are considered identical under all possible inflections.

"The man and the dog", and

"The woman search the dog"

TABLE 22. Four most preferable parses from the incomplete sentence "The XXX see the dog"

THE	MAN	SEE	THE	DOG
DET	N	V	DET	N
S	S	S	S	S
S	S	S	S	S
NP	NP	VP	VP	VP
E	E	E	NP	NP
E	E	E	E	E
E	E	E	E	E

THE	WOMAN	SEE	THE	DOG
DET	N	V	DET	N
S	S	S	S	S
NP	NP	VP	VP	VP
NP	NP	E	NP	NP
E	E	E	E	E
E	E	E	E	E
E	E	E	E	E

THE	MAN	AND	THE	DOG
DET	N	CONJ	DET	N
S	S	S	S	S
NP	NP	NP	NP	VP
NP	NP	E	NP	NP
E	E	E	E	E
E	E	E	E	E
E	E	E	E	E

THE	WOMAN	SEARCH	THE	DOG
DET	N	V	DET	N
S	S	S	S	S
NP	NP	VP	VP	VP
NP	NP	E	NP	NP
E	E	E	E	E
E	E	E	E	E
E	E	E	E	E

In the following table, a sentence is parsed with two corrupted words: "The XXX YYY the dog".

TABLE 23. Four most preferable parses of the incomplete sentence: "The XXX YYY the dog"

THE	MAN	AND	THE	DOG
DET	N	CONJ	DET	N
S	S	S	S	S
NP	NP	NP	NP	VP
NP	NP	E	NP	NP
E	E	E	E	E
E	E	E	E	E
E	E	E	E	E

THE	WOMAN	SEARCH	THE	DOG
DET	N	V	DET	N
S	S	S	S	S
NP	NP	VP	VP	VP
NP	NP	E	NP	NP
E	E	E	E	E
E	E	E	E	E
E	E	E	E	E

THE	WOMAN	SEARCH	THE	CAT
DET	N	V	DET	N
S	S	S	S	S
NP	NP	VP	VP	NP
NP	NP	E	NP	NP
E	E	E	E	E
E	E	E	E	E
E	E	E	E	E

THE	WOMAN	SEARCH	THE	WOMAN
DET	N	V	DET	N
S	S	S	S	S
NP	NP	VP	VP	VP
NP	NP	E	NP	NP
E	E	E	E	E
E	E	E	E	E
E	E	E	E	E

The model correctly returns the following possible parses in order of preference:

"The man and the dog..."

"The woman search the dog"

"The woman search the cat"

"The man beat the woman"

Most interesting in this case is the return of an incomplete sentence: "The man and the dog...". So not even in the case of wrongly spelled words, but also in the case of half or incomplete sentences, the model can derive the most probable parse. The fact that the model is able to do so, makes is possible to use this technique in many more applications where incomplete sentences occur. As the amount of context increases, it will be possible to complete this sentence, given the information in the corpus.

Both examples generated analyses with new words in the parsed sentence, substituted for the ones in the original sentence. For instance, "The XXX see the dog: is transformed into "The man and the dog". Moreover, "The XXX YYY the dog" is transformed into more preferable sentences (according to the corpus) such as: "The woman search the cat". Both are very interesting cases. In the first, the entire sentence is changed to a more preferable one. In the second "dog" is replaced by another animal, "cat". This is in fact a semantical substitution. This kind of "creative" behavior is a direct result of the corpus based approach and can never be observed in traditional rule based systems. Fortunately, the amount of creativity can be determined by a set of threshold values. High thresholds yield low creative behavior, low thresholds leads to creative behavior and may eventually lead to over-generalization.

7.3 Results Corpus 2: Case Role Assignments

The corpus trained with words, word categories, non-terminal structures and case role assignments appeared to be slightly better in the structural retrieval. However, the quantitative improvements are small. As the corpus 1 representation was almost always able to derive at least one proper structural analyses and some wrong ones (which were lower ranked), the corpus 2 representation derived more correct structures in the lower ranks. However, the percentage of correct hits for the higher ranked analyses was about the same.

However, the main advantage of this representation over that of type 1 is the ability of the model to process sentences like "Give the man the book", without considering "... the man the book" as one NP instead of two separated NP's. See Table 24, "Parse of "Give the man the book"" for details.

TABLE 24. Parse of "Give the man the book"

GIVE	THE	MAN	THE	BOOK
V	DET	N	DET	N
S	S	S	S	S
SEM	SEM	SEM	SEM	SEM
VP	NP	NP	NP	NP
SEM	GOAL	GOAL	THEME	THEME
E	E	E	E	E
SEM	SEM	SEM	SEM	SEM
E	E	E	E	E
SEM	SEM	SEM	SEM	SEM
E	E	E	E	E
SEM	SEM	SEM	SEM	SEM
E	E	E	E	E
SEM	SEM	SEM	SEM	SEM

Most interesting from this result is *not* the fact that the corpus is of a (slightly) higher quality, or that sentences such as above can be parsed properly. Much more impact makes the fact that we are able to add different types of knowledge, such as semantic tags without too much trouble in a very straightforward manner. Moreover, this knowledge is directly integrated from the highest to the lowest levels in the decision process. These properties underline the advantages of neural networks and mathematical methods in general for these knowledge integration applications. In addition, one might even consider adding other knowledge sources to the model as it is so simple to implement.

In a second simulation, the model was trained with an adapted corpus where only the ambiguous sentences where tagged with additional features to distinguish them from each other. The results were about just as good as the ones with a completely tagged corpus. On the one hand, this behavior was caused by the limited size and complexity of the corpus, so one can easily distinguish ambiguities. On the other hand, each additional feature adds something to the weight vector. For those sentences where there is no difference in the additional information, it will not add any additional distinguishing power.

7.4 Results Corpus 3: Non-Terminal Training

The inclusion of non-terminals in the training phase did increase the parsing results significantly. Below, the results of a simple parse can be found.

TABLE 25. Pars of "The man and the woman search the dog"

THE	MAN	AND	THE	WOMAN	SEARCH	THE	DOG
DET	N	CONJ	DET	N	V	DET	N
SEM	SEM	SEM	SEM	SEM	SEM	SEM	SEM
S	S	S	S	S	S	S	S
AGENT	AGENT	AGENT	AGENT	AGENT	SEM	SEM	SEM
NP	NP	NP	NP	NP	VP	VP	VP
SEM	SEM	SEM	SEM	SEM	SEM	THEME	THEME
NP	NP	E	NP	NP	E	NP	NP
SEM	SEM	SEM	SEM	SEM	SEM	SEM	SEM
E	E	E	E	E	E	E	E
SEM	SEM	SEM	SEM	SEM	SEM	SEM	SEM
E	E	E	E	E	E	E	E
SEM	SEM	SEM	SEM	SEM	SEM	SEM	SEM
E	E	E	E	E	E	E	E

In order to fully exploit the fact that non-terminals were included in the training process, a number of extensions were made to the original parsing process. If the results of a sentence analysis are not good enough, slots of adjacent words are replaced by a (smaller) number of empty slots. E.g., the string <ABCD> is replaced by <XBC>, <AXD>, <ABX>, and <XY>. Next, these sentence mutations are analyzed in the same manner as the initial sentence. On the position of the X and Y, terminals as well as non-terminals can occur. In the following table, an example of the evaluation of such an incomplete sentence can be found.

TABLE 26. Parse of the incomplete sentence "XXX search the dog"

NP	SEARCH	THE	DOG
NP	V	DET	N
SEM	SEM	SEM	SEM
S	S	S	S
AGENT	SEM	SEM	SEM
NP	VP	VP	VP
SEM	SEM	THEME	THEME
E	E	NP	NP
SEM	SEM	SEM	SEM
E	E	E	E
SEM	SEM	SEM	SEM
E	E	E	E
SEM	SEM	SEM	SEM
E	E	E	E

According to the table, the first slot is an NP, acting as an agent in the total sentence. Next, all possible NP's must be derived from the corpus by using a random search method. An NP can simply be extracted from the corpus by using an adjacent string of empty slots and setting an NP/Agent signal on the structure tags of this empty string. The parsing algorithm will then generate as many NP's as one wishes. By including a random method to evaluate the NP's generated, the model can fill the NP slot from Table 26, "Parse of the incomplete sentence "XXX search the dog"" with all kinds of different NP's.

Next, the sentences must be ranked in order of preference. Without the non-terminal evaluation, this could be done on the basis of results of the postprocessing phase. However, such an evaluation would eliminate the occurrence of new structural types not explicitly present in the corpus. So a different ranking algorithm is used. The NP as well as the mutated sentence will have an evaluation value derived from the corpus. These two values can be combined by multiplying them. If one of the two is bad, it will result in a bad result. However, if both are good, the overall result will be good.

8.0 Discussion

8.1 Comparisons of the Data Oriented Parsing Model to Other Parsing Models

The research that has been presented in this chapter has more the character of an interesting feasibility experiment than of a well balanced scientific evaluation of the (neural) data oriented parsing model. Therefore, one should be careful before comparing DOP with other parsing paradigms (or for comparing the current DOP at all with other paradigms) for a number of reasons:

- The corpus that has been used is very limited to yield any hard conclusions.
- As a result, the simulation results are more qualitative instead of quantitative.
- The DOP paradigm has not been worked out completely yet.
- The DOP paradigm is very new. As a result, the neural implementation was the first one that has been realized. The implementation that uses statistical techniques is in an even more premature state.

Nevertheless, the results that have been achieved here are extremely encouraging to continue the DOP project as a new direction in linguistic analysis that is indeed capable to be implemented in applications such as speech recognition, optical character recognition and information retrieval.

8.2 Data Oriented Parsing versus Neural Data Oriented Parsing

How closely does Neural Data Oriented Parsing (NDOP) follow the original Data Oriented Parsing (DOP) algorithm? An initial observation that can be made is that DOP works with a corpus of sentence fragments of different sizes, where NDOP uses a corpus with sentence fragments of equal size. Next, in DOP the notion of a structure is represented as a symbolic one, where NDOP represents structures as vectors of matrices. Most differences are consequences caused by the results of these differences in data representation.

The DOP philosophy may use conditional probabilities, information theoretic values or other statistical notions, where the NDOP uses the values derived by the Kohonen feature map. In section 7.5 "Information Theory" it is argued that these values closely resemble the information theoretic values as introduced by Shannon.

In other aspects, both techniques have much in common. Especially the addition of non-terminal training and case role assignment to NDOP made the structural difference between these two approaches smaller.

8.3 Structured Corpora

The role of the structure is three-fold.

- First there is the increase of quality of the corpus on the terminal and category level due to the addition of structure.
- Next, nonsense structures are eliminated by feeding forward the retrieved structures once again into the neural net to determine whether or not they are valid.
- Most important is the possibility to incorporate windows of non-terminals in the training process. By doing so, long term relations can be stored more easily in the corpus. Then, the system will probably suffer less from typical Markovian problems such as the lack of memory with smaller windows and the over-specification with larger window sizes.

The main problem with the use of structured corpora is the fact that they might not be available in large enough sizes. This has been true until recently. A corpus currently built at the University of Nijmegen is the largest one in Europe known to us. It consists of 1.5 million words. Other corpora of comparable sizes are also available.

8.4 Ambiguity

Probabilistic grammars show considerable problems in solving large ambiguities, even if the probabilities are set by using an efficient re-estimation technique such as used for Stochastic Context-Free Grammars. By using the Kohonen feature map as a corpus, the ambiguity is solved by the inherent features of the Kohonen map (or the inherent features of neural nets in general). The neural model solves ambiguity by matching all possible candidates in their proper context and returning the most probable one. Closely related concepts can be found immediately by scanning the direct neighborhood of the best neuron, making it possible to implement a disambiguation task very efficiently. Moreover, corrupted input or incomplete sentences can be processed too. The neural net shall return the most closely related fragment present in the corpus.

8.5 Kohonen Feature Maps

The presented model converges properly to an organization of the feature map in which all related objects are in neighboring areas. By studying slices of the feature map, it can be observed that this property holds for all elements of the window and all features within one window such as word, word category, structure, and semantical tags. Training times are approximately 100.000 up to 150.000 cycles for a corpus of 76 sentences. Larger corpora (up to 1.000 sentences) require comparable training times, although the absolute time may be longer due to an increase in the required neurons. The amount of required neurons

is about the number of different segments that occur in the corpus. In the corpus used, this was approximately 250. By using feature maps of 20 by 12 neurons, almost all segments could be represented.

However, the scalability of the model remains a problem. Kohonen feature maps tend to entangle when they are getting larger. As a result, neighboring neurons represent totally different objects. One method to avoid entangling is to train in a more context-sensitive way.

In [Kohonen, 1991], such a context sensitive training method is proposed: the *Hypermap*. This model is similar to the training algorithm presented here, however it trains global context properties first. As the model converges to a self-organizing map of contexts, objects are trained to the map to obtain a fine-grained grid where necessary. By incorporating such a training method, the results as presented in this research are expected to improve also. A first simulation indicated that larger corpora (300 sentences) could be trained by using this method.

Other attempts to increase the feature map capacity can be found in [Kangas, 1992] and in the studies of hierarchical feature maps as mentioned in the previous two chapters. However, most results in this direction are still immature. Another, more promising solution can be found in [Fritzke, 1992a-b], who grows his feature maps from small to large in such a manner that the feature maps are always in order. As a result, the neighborhood effects during the training will be less strong as they are in the original feature maps, but this can solved by adapting the simulations. More on the importance of the capacity of the feature maps can be found in the next section.

8.6 Information Theory

Much has been written on the relation between neural networks and Shannon's information theory [Shannon, 1948, 1949]. A neural network can be regarded as a (multi-stage)[1] encoder. Therefore, there is an obvious link between neural networks and information theory. In fact, notions from information theory can be used as a measure of performance of the neural network. More on these applications of information theory in neural network research can be found in [Bichsel et al., 1989], and [Kühnel et al., 1990].

However, information theory can also be used to explain other aspects of neural networks, such as the values, meaning, and relation with the natural sources of the sensory input. One of the first persons to do so was Ralph Linsker [Linsker, 1987, 1988, 1989a, 1989b, 1990][2]. In biological neural networks, sensory information is processed in a very efficient

1. In the case of a multi-layer feed forward network.

manner. Much of the natural data that is being processed is redundant. However, the human brain manages to remove most redundancies in an environment full of noise without the loss of information.

This notion is called the principle of *maximum information and minimum entropy*[1]. Linsker showed that real Hebbian learning (see chapter 1: "Neural Computation") implements redundancy elimination without loss of information. He calls this the *infomax* principle, predicting the maximization of the amount of information in the system.

Say L is an input signal vector and M is an output signal vector, then the neural network implements a mapping f: $L \rightarrow M$. $P(M|L)$ holds the conditional probability density function. The input probability density function is represented by $P_L(L)$ for the case there is no feedback from $M \rightarrow L$, the output probability density function by $P_M(M)$. Then, in order to maximize the total information R, the following equation needs to be maximized:

$$R = \sum_L \sum_M P_L(L) \cdot \log\left(\frac{P(M|L)}{P_M(M)}\right) \qquad \text{(EQ 7)}$$

Next, presume that the points in M are distributed equally, and they are in topographic order. Then, if the neural network follows the infomax principle, the maps that are formed will have the following properties (see [Linsker, 1987] for details):

- The areas representing M will be equally large and uniformly distributed over the feature map.
- The map will conserve the topological distribution of the underlying probability density function.

Closely related to the original Hebb rule is the Kohonen feature map. On the one hand, Kohonen feature maps (empirically) tend to have the same properties as maps produced according to the infomax principle. On the other hand, some of the quantitative principles needed for the infomax principle such as lateral interconnections and topographic order are also present in the Kohonen feature map algorithm.

So, there is reason to assume that the weights which are derived in the Kohonen feature maps are in fact values representing the properties of the sensor values in such a case that

2. An earlier application of information theory in neural networks can be found in [Gardner, 1986]. However, his work was not as influential as that of Linsker.

1. Shannon used the word entropy to indicate the measure of disinformation in a system. This term was suggested to him by John von Neumann as it would give him an edge in discussions about information theory because nobody else used the word entropy in communication theory.

the redundancy is eliminated without loss of information value. It is not stated that the weights in a Kohonen feature map *are* information theoretical values as proposed by Shannon and Linsker. The Kohonen model is much too limited with respect to the facts that:

- It does not maximize an expression such as the one given in the beginning of this section. Actually, it does not refer to any notion of information whatsoever.
- It is a local algorithm due to the local lateral interactions between the neurons. In the infomax principle, global interaction is presumed.
- It does not consider aspects such as noise.

But, there is a clear relation between the two models, indicating that the Kohonen feature map does converge to something more valuable than plain frequencies. Due to the complicated neighborhood interactions a set of weights is derived of which each represents an element of the training set with respect to the *frequency of occurrence as well as to the context in which it occurred*, just as it is the case in information theory.

So, Kohonen feature maps, as they are used in the simulations, derive conditional probabilities of the corpus fragments. Although one cannot give a quantitative measure of the amount of conditionality, it should be possible to indicate at least a lower and an upper bound of the context that is incorporated.

The Upper Bound

The conditionality is derived during the training process by the neighborhood effects. If the feature map is in perfect order (that is, all objects are organized in such a manner that related objects are in neighboring areas), the amount of conditionality will approximate the information theoretic values as indicated by Shannon because all elements are stored with respect to all the proper contextual influences that exist in the system.

The Lower Bound

However, for a perfect organization, Kohonen feature maps require that the form of the Kohonen feature map follows the form of the underlying probability distribution. Here, a two-dimensional, rectangular feature map is used, presuming that natural language is two-dimensional and rectangular. As all can see, this is a very rough simplification of the real world situation. Due to this limitation, and due to the fact that the feature map is not in perfect order during the training process, the values that are derived by the neurons do not incorporate all possible contextual influences and are therefore less good than information theoretic values. If one presumes that the feature map is never in perfect order, the values that are derived incorporate only the context as given by the shifting window: simple

Markovian conditionality, which can be considered as a lower bound of the value of the probabilities derived by the feature map.

So, the probabilities that are derived by the feature maps are better than Markovian conditional probabilities, but they are worse than Shannon information theoretic values. As a result, the more context sensitive and the less entangled the feature maps are during the training process, the better the probabilities are that they derive.

8.7 Computational Complexity

The model has different phases with different complexities. The computational complexity of the model has different dimensions. In the training phase, the model has a linear complexity in time and space with respect to the window size that is shifted over the text. However, the training time itself can still be quite long due to the large constant factors in the time complexity.

In the parsing phase, different rules hold. If one uses only adjacent fragments, then the number of partitions equals 2^{n-1} where n represents the number of words in a sentence. So, the number of partitions is exponential with respect to the number of words in a sentence. The number of fragments generated from a sentence of length n is given by the function $\frac{1}{2}n^2 + \frac{1}{2}n$. Once the fragments are generated, each of them has to be fed forward to the neural net. These are all constant complexity factors (although they get pretty high if the map size increases). However, all of the fragments as well as all of the neurons could be processed in parallel, making the system a powerful parsing device.

The phase in which the tree is checked on inconsistencies should not be taken too seriously. It is not necessary to process the entire graph. If a completely nonsensical structure is parsed, such as "the the the the", then all of these values will return errors that are higher than a certain threshold, resulting in infinitely large errors. Then none of the structures will be selected. The checking has more to do with direct neighbor interaction of structures. Depending on the representation of structure used, one uses more or less complicated post-processing techniques.

Once the neural net is trained, the complexity with respect to the number of words becomes an interesting issue. The total number of adjacent sentence fragments generated seemed to be quadratic, the total number of partitions exponential. Although these numbers seem to be pretty large, one cannot completely compare them to a chart parser because the model performs word and structure assignment as well as a disambiguation task.

Much can be optimized in this model such as:

- Most computational time is wasted on the determination of the maximum of the feature map. For all segments all n neurons must be evaluated. By deriving a binary tree which stores the weights of the neurons, the number of calculations is brought down to log n for each segment. Given the fact that 80% of the calculation time is used on this task and the fact that 300 neurons are used, an overall speed up of factor 7 is obtained.

- Currently, all partitions are evaluated. Here, a method can be used which selects random partitions. This is mainly possible due to the fact that the best parse can be derived in many different manners. In the simulations, most of the analyzed parses occurred more than once; the best parse occurred very often.

- By increasing the thresholds used, the number of partitions, fragments or segments to be evaluated can be reduced significantly. However, whether this results in a random effect or that only particular types of structures are eliminated is not known yet.

If all partitions are evaluated and the thresholds are set low, one can see that all possible (valid) fragment combinations present in the corpus will be generated (as with all corpus based parsing techniques, the knowledge representation is in the corpus, so the limits of the corpus are the limits of what is known in the system). However, if random search techniques are used and not all possibilities are evaluated, the probability that a possible sentence is not discovered gets larger.

8.8 Future Research

Future research can be divided into different directions:

- First, a sequential data-oriented parsing algorithm is under development. When this model is implemented, an interesting comparison between neural and statistical models can be made. This model could be used in cooperation with the type 3 corpus representation. The neural network then presents possible non-terminal structures, which can be then be replaced by complete NP's selected from the corpus by a statistical (Monte Carlo) method. By then, it may also be possible to implement larger DOP models, making it possible to compare DOP to other methods in parsing.

- Next, the Kohonen model can be upgraded along different paths. The recently introduced Hypermap training algorithm is expected to be much better in contextual representations of data. Other directions are the context sensitive map from Kangas or the growing structures from Fritzke. Moreover, one could represent the structure in a more subtle fashion. Instead of using a flat coding system, the structure could be represented by a hierarchical set of feature maps.

- It was shown that the model can be extended very easily with other knowledge types. This enables one to add e.g. pragmatic, discourse and other semantic tags to the structured corpus which disambiguate those cases in which additional disambiguation is needed.In the long run, other types of media such as pictures, sound and video are also possible candidates for corpus extensions.

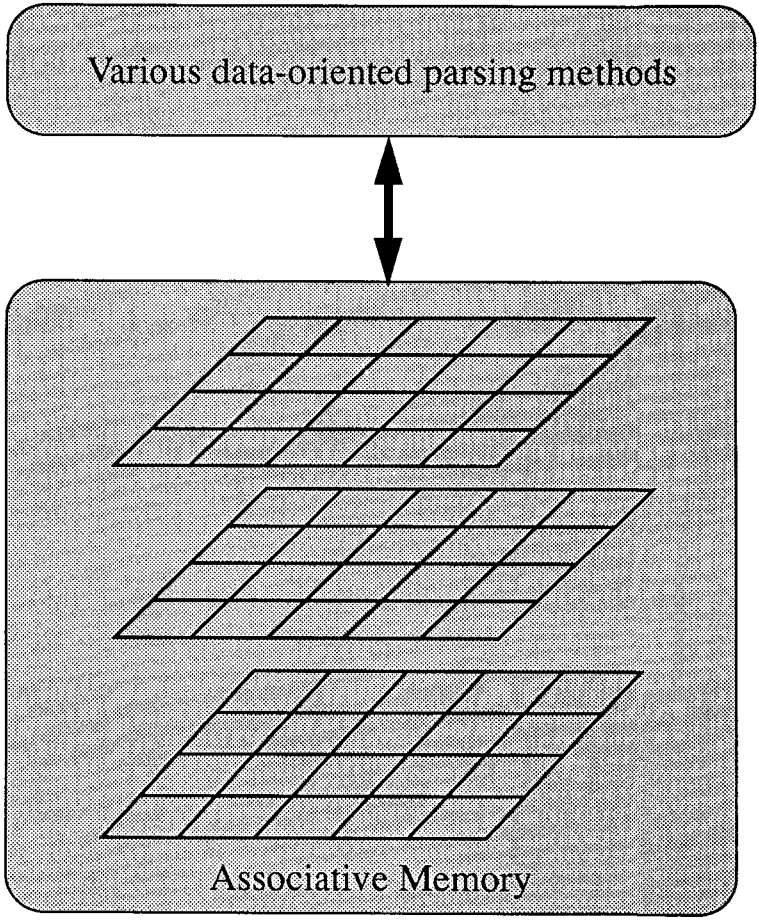


FIGURE 5. The neural data-oriented parsing model as an associative memory

- Finally, various new data-oriented parsing methods could be tried on the corpus. In the neural data-oriented parsing model, the Kohonen feature map is used as an associative memory on which more than one parsing algorithm could be tested (see (see figure 5, "The neural data-oriented parsing model as an associative memory").

Chapter 6

Discussion and Conclusions

"The only good data is more data"
-- Robert Mercer

Abstract

In this chapter, the objectives of the first two chapters shall be discussed with respect to the solutions and models presented in the chapters three to five. In order to increase the readability, this chapter is divided into different sections, handling different topics.

First, the relation of the work presented here with the field of Machine Intelligence (MI) is presented. The emphasis will be on the role of neural networks and Kohonen feature maps in particular. Next, the problems that occurred in the study of NLP are discussed in relation to the neural data oriented models. Then, the study of Information Retrieval (IR) is treated in a similar manner. Hereafter the relation with other fields such as autonomous, real-time, and expert systems is given. And, last but not least, the reader is provided with some directions for future research.

1.0 Machine Intelligence

Both traditional Artificial Intelligence (AI) and Neural Networks (NN) have their appeals and their drawbacks. Here, it is not the intention to compare AI with NN because they can hardly be compared objectively. On the one hand, there are tasks in which AI is simply better, such as the representation of hierarchical structures or the implementation of recursion. On the other hand, many of the problems that occur in traditional AI can better be solved with NN. As it appears, AI and NN are often complementary; if one of the two is successful, the other is probably not. Therefore, comparisons of the two approaches for a specific task are often not fair, because one of them probably suits the application better than the other.

More interesting is the relation of MI with the field of (statistical) Pattern Recognition (PR). This study has traditionally been concerned with the classification of objects through (low level) signal processing. Over time, only statistical techniques appeared to be relevant. Structural pattern recognition (the AI branch of pattern recognition) failed already many years ago. Many in the field of PR argue that neural networks are nothing else than a fancy method of implementing a (not well understood) statistical classification technique.

It is a fact that Hidden Markov Models (HMM) and other statistical re-estimation techniques have much in common with methods used in neural networks. But, there remain a number of significant differences between the sequential statistical algorithms and neural network technology:

- Most of the statistical PR models implement supervised function approximators. Self-organizing models do exists, but are not as good as they are in neural network research.
- During the training process, self-organizing neural networks show very interesting recurrent interactions, which enable the models to derive organizations and classifications that are based on more than just simple adjacent context dependencies. Kohonen feature maps implement such neighborhood effects.
- Generalization and association are implicitly present in the neural network models.

Of course, one could always implement this typical neural behavior in a sequential statistical algorithm, but then, one has to program these features explicitly one by one (which can be quite a difficult task).

On the other hand, there are some major drawbacks of neural networks that currently limit their full application in the real world.

- If one applies neural technology to an application, it either works, or it does not work. It is impossible to *patch* the model for certain exceptions, which can be done in a sequential algorithm. This aspect is mainly due to the distributed data representations and the parallel character of neural networks. The first is responsible for the fact that one does

not know where to attach the patch (many neurons are representing many concepts). The latter causes a complicated process in which it is hard to say which neuron is responsible for which action (the credit assignment problem).

- In addition, neural technology is not really scalable. Almost all experiments are based on sequential simulations of parallel algorithms. Parallel hardware is rare and very expensive. Moreover, many of the models tend to collapse if they are applied to large problems.

Statistical pattern recognition does have some proper answers to both problems, making it more suited in many real-world applications. In particular if it uses re-estimation techniques such as is the case in speech recognition with HMM.

1.1 Kohonen Feature Maps

Because all models used in this research were based on the Kohonen feature map, some additional remarks are made on this subject.

In this work, neural networks are used for different tasks. In the recurrent Kohonen feature maps, they are used for the processing of language. In the Neural Filter, they are used to derive a contextual representation of some query, and in the Neural Data-Oriented Parsing model they are used to derive an internal representation of the corpus. The first application explicitly implemented language processing, where the latter two used the feature maps as an associative memory. The processing was carried out by an external sequential algorithm.

As it appeared, the results indicate that feature maps implement finite state behavior in the recurrent model, although stability and convergence speed were not among the virtues of the system. As finite state behavior is not enough for natural language processing and because the representation of hierarchical structures requires large amounts of neurons (there is evidence that the number of neurons grows exponentially with the number of states, [Servan-Schreiber et al., 1991]), the application of such models is limited. As a matter of fact, many of the language processing models as they exist in the back-propagation community suffer from similar problems.

The neural network behaved much better when it was used as an associative memory of a larger system rather than as a processing device. But there are still a number of problems with the feature maps:

- It is difficult to use the models in larger applications. As soon as the feature maps grow larger, they tend to entangle. One of the reasons for this behavior is the fact that larger maps have less boundary effects (which are very important for the organization process). In order to avoid this, one must try to keep the feature maps as much in order as

possible during the training process. There are a number of methods to do so. First, one can try to combine many smaller maps into a hierarchical system. Secondly, one can use more context sensitive training algorithms such as the *Hypermap* or the models presented by Kangas. Another option is the use of growing models as introduced by Martinetz and Fritzke. The feature maps of the latter two are always in order because they start with a restricted set of neurons and only add new neurons if they are needed.

- The feature maps exhibit interesting neighborhood effects. Due to these effects, contextual relations are incorporated in the derivation of the feature map weights. However, it is not completely clear how much context is incorporated in this process. The Kohonen feature map implements "a very complicated process" [1] that cannot easily be described quantitatively. However, some qualitative indications can be given. It can be argued that the better the form of the feature map adopts the form of the underlaying probability distribution, the better the model incorporates contextual relations in the derived conditional probabilities. In addition, the more the map is in order during the training process, the better the derived probabilities will be. In Chapter 5, it is argued that a lower bound for those values is an adjacent Markovian conditional probability and that an upper bound is a Shannon information theoretical value.

These two points are much interrelated. If the maps entangle less, the map is of higher quality. Although this issue is of great interest to the community, none of the experts in the Kohonen feature maps can provide quantitate measurements on the quality of the feature maps. Moreover, much effort is invested in better scalable models. If work in this direction is not successful, the Kohonen feature maps may encounter tough times.

1.2 Knowledge Representation & Associative Memories

Here, Kohonen feature maps are used as an associative knowledge representation scheme. They are nothing more than an interesting context sensitive representation method. However, much of the work presented here can also be implemented by using other knowledge representations. But the question is whether they will be as good.

The most impressive feature of the Kohonen maps is the automatic derivation of a topological map in such a manner that all related objects are stored in neighboring areas. Moreover, the feature map derives conditional probabilities by incorporating much more context than simple Markovian features. When implementing other knowledge representation models, it will be a challenge to incorporate these two properties in the system, as they are mainly responsible for the required behavior.

1. Personal communication with Professor T. Kohonen.

2.0 Natural Language Processing

In this section, two issues are discussed in separated subsections. First, the implications of the results of the application of neural networks to natural language processing (NLP), and next the results of the usage of data oriented parsing (DOP) techniques in NLP.

2.1 NLP and Neural Networks

As it was argued in the opening chapters, traditional AI is not capable of addressing all the problems that occur in natural language processing. So, are these problems attacked successfully by neural techniques? The answer to this question is twofold.

- On the one hand, neural networks are able to integrate knowledge from different sources, thereby disambiguating language more easily. The data oriented parsing model successfully integrated word, syntax category, syntax structure and case role information. Other additions to the model can be implemented easily. Moreover, neural networks are able to process corrupted sentences and they can even correct them to proper sentences by using the generalization capabilities of the neural network. It can even be argued that the DOP model shows creative behavior as it replaces words by more probable ones within that context. In addition, all information that was stored in the Kohonen models in the Chapters 3 to 5 was derived automatically.
- On the other hand, neural networks significantly restrict the processing possibilities by their inability to implement (dynamic) hierarchical structures. Neural networks can only process large vectors. So, hierarchical structures (as they appear in natural language) need to be transformed into vectors in such a manner that the vectors can also be re-transformed into hierarchical structures. In the DOP models, this has been done by using a model that restricts the number of hierarchical levels in the structure of the corpus to a maximum of six.

So, this research showed that neural networks are able to implement robust behavior, commensense generalizations and disambiguation on the basis of different knowledge sources. Most responsible for this behavior is the vector base on which the internal data representation is founded. Here, a possible solution to this problem has been given. Albeit not a very elegant one. Maybe that future research in this direction can provide a better transformation method between hierarchical structures and vectors.

However, most limiting feature of neural networks is the structural problem in up-scaling the technology. Due to conceptual as well as practical reasons, standard Kohonen feature maps are not suited for massive parallelism. As soon as the number of neurons exceeds a couple of thousand, the model does not converge as nicely anymore.

2.2 Data Oriented Parsing

Besides the theoretical issues of the application of neural networks to NLP as discussed in the previous sub-section, there is a more practical one. In the DOP model, language is processed on the basis of comparisons with a large collection (corpus) of analyzed examples. This in contrast to the traditional computational linguistic models which analyze language by inferencing over grammar rules. The examples in the corpus were all stored in their natural context (that is, examples that were related to each other, given some features, are stored in neighboring areas of the memory). As it appeared, the model was able to process new sentences, corrupted sentences and also showed some creative behavior. This in contrast to traditional rule based systems.

Here, it is argued that corpus based systems are more suited to implement successful NLP than rule based systems. Generative grammars, as they are used to process language can never cover the entire set of sentences in a language. There will always be exceptions and new additions. Moreover, the grammars will become so complicated, that one will no longer be able to maintain or extend them.

Language is presumed to be a consistent system. But, as it appears, many utterances in natural language are irregular and not subject to any rules. Therefore, natural language can better be described by a large set of examples. Moreover, there are obvious statistical influences that can be observed in language processing. DOP incorporates these statistical properties elegantly in a structural framework.

Most of the early corpus-based systems ignored the fact that language has hierarchical structures and shows (restricted?) recursive behavior. Therefore, these systems cannot process much of the commonly used language without adding every utterance to the corpus (which is of course nonsense). Here, as many utterances as possible (or as there are available) are added to the corpus, together with their structure and meaning, resulting in a system that can process old as well as new sentences with respect to the information in the corpus.

One of the most interesting features of the neural DOP model is the fact that all sentence fragments are stored in the corpus in relation to the rest of the corpus. This is, every sentence fragment is positioned in a related neighborhood on the feature map and every sentence fragment is labelled with a conditional probability that is conditional with respect to the rest of the corpus (see Section 1.0 on page 244 for an elaboration on this subject). This property enables the system to generalize over all features, resulting in the "creative" bevavior as reported.

3.0 Information Retrieval

Two different types of information retrieval (IR) problems can be distinguished. On the one hand there are relatively static data collections, also known as document retrieval systems. On the other hand there is the real-time filtering problem with relatively dynamic data collections. Although both applications differ in approach, they have one similar problem: the large data collections do not enable the use of any other method than a global surface analysis. Here the emphasis has been on the filtering problem; however, the same principles may be applied to the document retrieval problem.

In order to incorporate more context, a neural net is used to derive an internal representation of a certain interest. As it is argued, this self-organizing neural net derives a map of conditional probabilities that are somewhere in between simple Markovian windows and Shannon information theoretical values. By doing so, the filter incorporates more contextual information in the filtering process than models that are based on adjacent n-gram analysis. All this without loss of speed.

The relation between information retrieval and corpus based parsing is much closer than one may suspect on first sight. In IR, the size of the data collections is the main reason for the (practical) limitations of availible procesing techniques. There is simply no time to analyse text according to some rule based system. Results must be retrieved instantaneously. So, the only realizable manner to incorporate structural information in information retrieval is by tagging sentences in the corpus with structural features, just as in the data oriented parsing (DOP) paradigm. Next, this structural information can be used in the retrieval phase.

But there is another relation between IR and DOP. As the DOP corpora grow larger and larger, retrieval of relevant sentences becomes a problem. It is exactly here that IR provides a solution by incorporating advanced indexing and retrieval algorithms. Moreover, as IR has been confronted with large data collections before, data maintenance and compressing algorithms can be useful in DOP too.

It is this relation that links IR and DOP. On the one hand, document retrieval systems can become more context sensitive by using corpus based linguistic techniques. On the other hand, DOP can use IR techniques to maintain and control larger corpora.

4.0 Relation to other Studies

So, what is the relation of the models and results presented here to other studies? First, there is a clear relation to other models in machine intelligence, natural language processing and information retrieval (see chapter 1 and 2 for an overview). Here, a few other, less obvious, relations shall be discussed in random order.

- In *real-time systems*, there is no time to inference over rules in order to derive a conclusion[1]. Solutions must be retrieved instantaneously. The only way to implement such system behavior is by storing possible solutions in a data base. The techniques that are used in information retrieval as well as in data oriented parsing are then useful.
- Moreover, one of the insights gained throughout this research, is that it is much more sensible to design a computer system that confronts a moderate user with the results of its actions, than to build an *expert computer* that takes all decisions on its own and then confronts a non-knowledgable user with the results of it. This result can be applied to the field of natural language processing as well as to that of information retrieval
- An interesting combinations between reasoning, natural language processing and information retrieval may be in the study of *case based reasoning* (CBR). By using (natural language) scripts that describe solutions to problems that occurred previously, and tagging those scripts with structural information on the reasoning process, an alternative to expert systems can be provided. The relation of this approach to DOP may be clear.

1. In theory, real-time parsing is possible with a rule based system. However, in practice is is not and it is still a question if it will be ever fast enough.

5.0 Future Research

In this section a number of possible directions for future results shall be presented.

5.1 Data Oriented Parsing

Most future research can be found in the data oriented parsing (DOP) domain, as it is here where the best results were obtained. A number of possible extensions of the DOP model are:

- Implement larger corpora.
- Add more contextual and structural information to the corpus such as discourse.
- Use different parsing techniques.
- Develop other applications in which the DOP formalism can be used such as speech recognition, optical character recognition and case based reasoning.

5.2 Kohonen Feature Maps

The Kohonen feature maps were mainly responsible for the limited scalability of the models. This is caused by a number of reasons:

- Kohonen feature maps are not really time sensitive
- Kohonen feature maps need boundary effects in order to derive meaningful feature maps. Larger feature maps simply have relatively less boundaries and therefore derive qualitatively less well maps.
- It is hard to combine multiple feature maps in an elegant manner (although much research has been devoted to this subject).
- There is no (cheap) parallel hardware to implement Kohonen feature maps yet.

Extensions of the Kohonen model that address these problems should have high priority in order to use the maps in practical models.

5.3 Evaluation of Statistical Methods

As neural nets still show considerable drawbacks for real-world applications and as solutions for these problems are not to appear shortly, it may be wise to investigate other implementation techniques such as advanced statistical (re-estimation) techniques.

However, it shall be hard to beat the context sensitivity that is implicitly implemented in the Kohonen models.

References

[Aho et al., 1972]: Aho, A.V. and Ullman, J.D. (1972). *The Theory of Parsing, Translation and Compiling. Vol. 1: Parsing*. Prentice-Hall.

[Allen, 1990]: Allen, R.B. (1990). Connectionist Language Users. *Connection Science*, Vol. 2, No. 4, pp. 279-312.

[Allen, 1991]: Allen, R.B. (1991). Knowledge Representation and Information Retrieval with Simple Recurrent Networks. *Working Notes of the AAAI SSS on Connectionist Natural Language Processing*. March 26-28, Palo Alto, CA., pp. 1-6.

[Amari, 1977]: Amari, S.-I. (1977). Neural Theory of Association and Concept-Formation. *Biological Cybernetics*, Vol. 26, pp. 175-185.

[Amari, 1988]: Amari, S.-I (1988). Mathematical Foundations of Neurocomputing. *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1443-1463.

[Anderson et al., 1988]: Anderson, J.A. and Rosenfeld, E. (Eds.) (1988). *Neurocomputing (Foundations of Research)*. MIT Press.

[Anderson et al., 1990]: Anderson, J.A., Pellionisz, A. and Rosenfeld, E. (Eds.) (1990). *Neurocomputing 2 (Directions for Research)*. MIT Press.

[Anderson, 1972]: Anderson, J.A. (1972). A Simple Neural Network Generating an Interactive Memory. *Mathematical Biosciences*, Vol. 14, pp. 197-220.

[Anderson, 1977]: Anderson, J. (1977). Induction of Augmented Transition Networks. *Cognitive Science*, Vol. 1, pp. 125-157.

[Anderson, 1983]: Anderson, J.A. (1983). Cognitive and Psychological Computation with Neural Models. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13, pp. 799-815.

[Arbib et al., 1979]: Arbib, M.A. and Caplan, D. (1979). Neurolinguistics Must Be Computational. *Behavioral and Brain Sciences*, Vol. 2, pp. 449-483.

[Arbib et al., 1990]: Arbib, M.A. and Robinsonb, J.A. (1990). *Natural and Artificial Parallel Computation*. MIT Press.

[Arbib, 1985]: Arbib, M.A. (1985). *In Search of the Person, Philosophical Explorations in Cognitive Science*. The University of Massachusetts Press.

[Arbib, 1987]: Arbib, M.A. (1987). *Brains, Machines, and Mathematics, 2nd Edition*. Springer-Verlag.

[Aristotle, ca. 400 B.C.]: Aristotle (ca. 400 B.C.). "De Memoria et Reminiscentia", *Aristotle on Memory*, Richard Sorabji (Translation). Brown University Press.

[Barrett, 1988]: Barrett, E. (Editor) (1988). *Text, Context and Hypertext*. MIT Press.

[Barrett, 1989]: Barrett, E. (Editor) (1989). *The Society of Text*. MIT Press.

[Baum et al., 1963]: Baum, L.E. and Eagon, J. (1963). An Inequality with Applications to Statistical Prediction for Functions of Markov Processes and to a Model for Ecology. *Bull. Amer. Math. Society*, Vol. 73, pp. 360-363.

[Bein et al., 1988]: Bein, J. and Smolensky, P. (1988). Applications of the Interactive Activation Model to Document Retrieval. *Technical Report* CU-CS-405-88, University of Colorado, Boulder, CO.

[Belew et al., 1988]: Belew, R.K. and Holland, M.P. (1988). A Computer System Designed to Support the Near-Library User of Information Retrieval. *Microcomputers for Information Management*, Vol. 5, No. 3, pp. 147-167.

[Belew, 1986]: Belew, R.K. (1986). Adaptive Information Retrieval: Machine Learning in Associative Networks. *Ph.D. Thesis*, Univ. Michigan, CS Department, Ann Harbor, MI.

[Belew, 1987]: Belew, R.K. (1987). A Connectionist Approach to Conceptual Information Retrieval. *Proceedings of the First International Conference on Artificial Intelligence and Law*, pp. 116-126. ACM Press.

[Belew, 1989]: Belew, R.K. (1989). Adaptive Information Retrieval: Using a Connectionist Representation to Retrieve and Learn About Documents. *Proceedings of the 12th ACM-SIGIR Conference on Research & Development in Informatrion Retrieval*. June 11-20. Cambridge, MA, pp. 11-20.

[Bernstein, 1981]: Bernstein, J. (1981). Profiles: AI, Marvin Minsky. *The New Yorker*, December 14, pp. 50-126.

[Bichsel et al., 1989]: Bichsel, M. and Seitz, P. (1989). Minimum Class Entropy: A Maximum Information Approach to Layered Networks. *Neural Networks*, Vol. 2, Nr. 2, pp. 133-141.

[Blair et al., 1985]: Blair, D.C. and Maron, M.E. (1985). An Evaluation of Retrieval Effectiveness for a Full-Text Document-Retrieval System. *Communications of the ACM*, Vol. 28, No. 3, pp. 289-299.

[Bochereau Laurent et al., 1991]: Bochereau Laurent (1991). Extracting Legal Knowledge by Means of a Multi Layer Neural Net. *Proceeding of the International Conference on Artificial Intelligence in Law,*

[Bod, 1992]: Bod, R. (1992). A Computational Model of Language Performance: Data Oriented Parsing. *Proceedings of the COLING-92*, Nantes, France.

[Boekee et al., 1988]: Boekee, D.E. and van der Lubbe (1988). *Informatietheorie* (in Dutch). Delftsche Uitgevers Maatschappij (DUM).

[Bokhari, 1981]: Bokhari, S.H. (1981). On the Mapping Problem. *IEEE Transactions on Computers*, Vol. 30, No. 3, pp. 207-214.

[Bounds, 1989]: Bounds, D. (1989). Expert Systems and Connectionist Networks. In: *Connectionism in Perspective* (R. Pfeiffer et al., Eds.), pp. 277-282. North-Holland.

[Bourlard et al., 1989]: Bourlard, H. and Wellekins, C.J. (1989). Links Between Markov Models and Multilayer Perceptrons. In: *Advances in Neural Information Processing Systems* (D.S. Touretzky, Editor), Vol. 1, pp. 502-510. Morgan Kaufmann.

[Bourlard et al., 1991]: Bourlard, H., Morgan, N. and Wooters, C. (1991). Connectionist Approaches to the Use of Markov Models for Speech Recognition. In: *Advances in Neural Information Processing Systems* (D.S. Touretzky, Editor), Vol 3, pp. 213-219. Morgan Kaufman.

[Boyer et al., 1977]: Boyer, R.S., Moore, J.S. (1977). A Fast String Searching Algorithm. *Communications of the ACM*, Vol. 20, No. 10, pp. 762-772.

[Bozinovic et al., 1982]: Bozinovic, R., Srihari, S.N. (1982). A String Correction Algorithm for Cursive Script Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Nov. 1982, pp. 655-663, pp. 236-244.

[Brachman et al., 1988]: Brachman, R.J. and McGuinness, D.L. (1988). Knowledge Representation, Connectionism, and Conceptual Retrieval. *Proceedings of the 11th ACM-SIGIR Conference on Research & Development in Information Rertieval*, pp. 161-174.

[Bradshaw et al., 1989]: Bradshaw, G., Fozzard, R. and Ceci, L. (1989). A Connectionist Expert System That Actually Works. In: *Advances in Neural Information Processing Systems* (D.S. Touretzky, Editor), Vol. 1, pp. 248-255. Morgan Kaufmann.

[Bridle, 1990]: Bridle, J.S. (1990). Alpha-nets: a Recurrent 'Neural' Network Architecture with a Hidden Markov Model Interpretation. *Speech Communication*, Vol. 9, pp. 83-92.

[Broca, 1865]: Broca, P. (1865). Sur le Siège de la Faculté du Language Articul. *Bull. Society Anthropol.*, Vol. 6, pp. 377-393.

[Brown et al., 1990a]: Brown, P.F., Della Pietra, S.A., Della Pietra, V.J., Lai, V.L., and Mercer, R.L. (1990). An Estimation of the Upper-Bound for the Entropy of English. *Technical Report*, IBM Research, New York, NY.

[Brown et al., 1990b]: Brown, P.F., Cocke, J., Della Pietra, S.A., Della Pietra, V.J., Jelinek, F., Lafferty, J.D., Mercer, R.L., and Roossin, P.S. (1990). A Statistical Approach to Machine Translation. *Computational Linguistics*, Vol. 16, No. 2, pp. 79-8

[Bryson et al., 1969/1975]: Bryson, A.E. and Ho, Y-C. (1969). *Applied Optimal Control.* Hemisphere Publishing.

[Buhman et al., 1988]: Buhman, J. and Schulten, K. (1988). Storing Sequences of Biased Patterns in Neural Networks with Stochastic Dynamics. In: *Neural Computers*, (R. Eckmiller and Chr. von der Malsburg, Eds), pp. 231-242. Springer-Verlag.

[Cajal, 1906]: Cajal, S.R. (1906). The Structure and Connexions of Neurons. In: *Nobel Lectures: Psysiology amd Medicine*, 1901-1921, pp. 220-253. Elsevier Science Publishers.

[Carpenter et al., 1988]: Carpenter, G.A. and Grossberg, S. (1988). The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network. *IEEE Computer, Special Issue on Artificial Neural Systems*, Vol. 21, nr. 3, pp. 77-88.

[Charniak, 1983]: Charniak, E. (1983). Passing Markers: A Theory of Contextual Influence in Language Comprehension. *Cognitive Science*, Vol. 7, pp. 171-190.

[Chen et al., 1991]: Giles, G.L., Chen, D., Miller, C.B., Chen, H.H., Sun, G.Z., Lee, Y.C., (1991). Second Order Recurrent Neural Networks for Grammatical Inference. *Proceedings of the International Joint Conference on Neural Networks*. July. 8-12. Seattle, WA, pp. II 273 - II 281.

[Chomsky, 1957]: Chomsky. N. (1957). *Syntactic Structures*. Mouton.

[Chomsky, 1965]: Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT Press.

[Chrisley, 1990]: Chrisley, R.L. (1990). Cognitive Map Construction and Use: A Parallel Distributed Processing Approach. In: *Connectionist Models: Proceedings of the 1990 Summer School* (D.S. Touretzky, J.L. Elman, T.J. Sejnowski and G.E. Hinton, Eds.), pp. 287-300. Morgan Kaufmann.

[Church, 1988]: Church, K.W. (1988). A Stochastic Parts Program an Noun Phrase Parser for Unrestricted Text. *Proceedings of the Second ACL Conference on Applied Language Processing*.

[Cleeremans et al., 1989]: Cleeremans, A., Servan-Schreiber, D. and McClelland, J.L. (1989). Finite State Automata and Simple Recurrent Networks. *Neural Computation*, Vol. 1, No. 3, pp. 372-381.

[Cleeremans et al., 1990]: Cleeremans, A., McClelland, J.L. (1990). Learning the Structure of Event Sequences. *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, July 25-28, Cambridge, MA, pp. 709-716.

[Cochet et al., 1988]: Cochet, Y. and Paget, G. (1988). ZZENN:ZIG ZAG Epigenetic Neural Networksand their use in Information Systems. *Proceedings of Neuro*, June 6-9, 1988, Paris, France, pp. 663-672.

[Cochet et al., 1988]: Cochet, Y. and Paget, G. (1988). ZZENN:A New Approach to Connectionist Machine Learning. *Proceedings of the International Computer Science Conference, Artificial Intelligence: Theory and Applications*, Hong Kong, pp. 377-380.

[Cohen et al., 1987]: Cohen, P.R. and Kjeldsen, R. (1987). Information Retrieval By Constrained Spreading Activation in Semantic Networks. *Information Processing & Management*, pp. 255-268.

[Cooper, 1971]: Cooper, W.S. (1971). A Definition of Relevance for Information Retrieval. *Information Storage and Retrieval*, Vol. 7, No. 1, pp. 19-37.

[Cotter et al, 1992]: Cotter, N.E. and Guillerm, T.J. (1992). The CMAC and a Theorem of Kolmogorov. *Neural Networks*, Vol. 5, pp. 221-228. Pergamon Press.

[Cottrell et al. 1990]: Cottrell, G.W. and Tsung, F. (1990). Learning Simple Arithmetic Procedures. In: *Advances in Connonnectionist and Neural Computation Theory: High Level Connectionist Models*(J.A. Barnden and J.B. Pollack Eds.), Vol. 1, pp. 305-321. Ablex Publishers.

[Cottrell et al., 1983]: Cottrell, G.W. and Small, S.L. (1983). A Connectionistic Scheme for Modeling Word-Sense Disambiguation. *Cognition and Brain Theory*, Vol. 6, nr. 1, pp. 89-120.

[Cottrell et al., 1989]: Cottrell, G.W. and Tsung, F. (1989). Learning Simple Arithmics Procedures. *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, August. 16-19. Ann Arbor, MI, pp. 58-65.

[Cottrell, 1985]: Cottrell, G.W. (1985). Connectionist Parsing. *Proceedings of the 7th Annual Conference of the Cognitive Science Society*, Irvine, IL, pp. 201-211.

[Cottrell, 1989]: Cottrell, G.W. (1989). *A Connectionistic Approach to Word Sense Disambiguation*. Morgan Kaufmann.

[Croft et al., 1979]: Croft, W.B., Harper, D.J. (1979). Using Probabilistic Models of Information Retrieval without Relevance Information. *Journal of Documentation*, Vol. 35, No. 4, pp. 285-295.

[Croft et al., 1983]: Croft, W.B., Wolf, R., Thompson, R. (1983). A Network Organization for Document Retrieval. *Proceedings of the 6th International ACM -SIGIR Conference on Research and Development in Information Retrieval*, pp. 178-188.

[Croft et al., 1991]: Croft, W.B., Turtle, H.R. and Lewis, D.D. (1991). The Use of Structured Queries in Information Retrieval. *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval.* October 13-16, Chicago, Illinois. pp. 32-45.

[Croft, 1977]: Croft, W.B. (1977). Clustering Large Files of Documents Using the Single Link Method. *Journal of the American Society for Information Science (JASIS)*, Vol. 28, No. 6, pp. 341-344.

[Croft, 1980]: Croft, W.B. (1980). A Model of Cluster Searching Based on Classification. *Information Systems*, Vol. 5, No. 3, pp. 189-195.

[Croft, 1981]: Croft, W.B. (1981). Document Representation in Probabilistic Models in Clustered Files. *Journal of the American Society for Information Science (JASIS)*, Vol. 31, No. 6, pp. 451-457.

[Crough, 1986]: Crough, D.B. (1986). The Visual Display of Information in an Information Retrieval Environment. *Proceedings of the 9th International ACM -SIGIR Conference on Research and Development in Information Retrieval*, pp. 58-67.

[D'Amore et al., 1988]: D'Amore, R. and Mah, C. (1988). N-Grams. *PAR Government Systems Corp.*, McLean, Virginia.

[Dealemans et al., 1992]: Daelemans, W. and Powers, D.M.W. (1992). Proceedings of the First SHOE Workshop. February 27-28, Tilburg University, The Netherlands.

[DARPA, 1988]: DARPA, Defence Advanced Research Program Agency (1988). *Darpa Neural Network Study*, October 1987-February 1988. Lincoln Laboratory, MIT.

[DARPA, 1991]: DARPA, Defense Advanced Research Project Agency (1991). Message Understanding Conference (MUC-3). *Proceedings Of the Third Message Understanding Conference (MUC-3)*, DARPA.

[Dell, 1985]: Dell, G.S. (1985). Positive Feedback in Hierarchical Models: Applications to Language Production. *Cognitive Science*, Vol. 9, pp. 3-23.

[Dell, 1986]: Dell, G.S. (1986). A Spreading Activation Theory of Retrieval in Sentence Production. *Psychological Review*, pp. 283-321.

[Dennis et al, 1991]: Dennis, S. and Phillips, S. (1991). Analysis Tools for Neural Networks. *Connectionist discussion list: connectionists@cs.cnu.edu*, July 24 1991.

[Devijver et al., 1982]: Devijver, P.A. and Kittler, J. (1982). *Pattern Recognition*, A Statistical Approach. Prentice Hall.

[Dolan et al., 1987]: Dolan, C.P. and Dyer, M.G. (1987). Symbolic Schemata, Role Binding, and the Evolution of Structure in Connectionist Memories. *Proceedings of the IEEE International Neural Network Conference*. June 21-24. San Diego, CA, pp. 287-298.

[Dolan et al., 1988]: Dolan, C.P. and Smolensky, P. (1988). Implementing a Connectionist Production System Using Tensor Products. In: *Proceedings of the 1988 Connectionist Summer School* (D.S. Touretzky, G.E. Hinton and T.J. Sejnowski, Eds.), pp. 265-274. Morgan Kaufmann.

[Dolan et al., 1989]: Dolan, C.P. and Smolensky, P. (1989). Tenson Product Production System: a Modular Architecture and Representation. *Connection Science*, Vol. 1, No. 1, pp. 53-68.

[Dorffner, 1988]: Dorffner, G. (1988). NETZSPRECH - Another case for Distributed 'Rule' Systems. *Proceedings of the 10th Annual Conference of the Cognitive Science Society*, August 17-19. Montreal, Canada, pp. 573-579.

[Dorffner, 1991]: Dorffner, G. (1991). "Radical" Connectionism for Natural Language Processing. *Working Notes of the AAAI Spring Symposium Series*, Palo Alto, CA, pp. 95-106.

[Doszkocs et al., 1990]: Doszkocs, T.E., Reggia, J. and Lin, X. (1990). Connectionist Models and Information Retrieval. *Annual Review of Information Science and Technology*, Vol. 25, pp. 209-260.

[Doyle, 1961]: Doyle, L.B. (1961). Semantic Road Maps for Literature Searchers. *Journal of the ACM*, Vol. 8, pp. 553-578.

[Duda et al., 1973]: Duda, R.O. and Hart, P.E. (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons.

[Dumais et al., 1988]: Dumais, S.T., Furnas, G.W., Landauer, T.K., Deerwater, S. and Harshman, R. (1988). Using Latent Semantic Analysis to Improve Access to Textual Information. *ACM, CHI'88*, pp. 281-285.

[Dyer, 1988]: Dyer, M.G. (1988). Symbolic NeuroEngineering for Natural Language Processing: A Multilevel Research Approach. *Technical Report*, TR UCLA-AI-88-14, UCLA, La Jolla, CA.

[Dyer, 1990a]: Dyer, M.G. (1990). Distributed Symbol Information and Processing in Connectionist Networks. *Journal of Experimental Artificial Intelligence*, Vol. 2, pp. 215-239.

[Dyer, 1990b]: Dyer, M.G. (1990). Symbolic NeuroEngineering for Natural Language Processing: A Multilevel Research Approach. In: *Advances in Connectionist and Neural Computation Theory: High Level Connectionist Models*(J.A. Barnden and J.B. Pollack Eds.), Vol. 1, pp. 32-86. Ablex Publishing.

[Edelman, 1987]: Edelman, G.M. (1987). *Neural Darwinism: The Theory of Neuronal Group Selection*. Basic Books.

[Eichmann et al., 1991]: Eichmann, D.A. and Srinivas, K. (1991). Neural Network-Based Retrieval from Reuse Repositories. *CHI '91 Workshop on Pattern Recognition and Neural Networks in Human-Computer Interaction*, April 28. New Orleans, LA.

[Eichmann et al., 1992]: Eichmann, D.A. and Srinivas, K. (1992). Neural Network-Based Retrieval from Reuse Repositories. In: *Neural Networks and Pattern Recognition in Human Computer Interaction* (R. Beale and J. Findlay, Eds.), Ellis Horwood.

[Eimas et al., 1990]: Eimas, P.D. and Galaburda, A.M. (Eds.) (1990). *Neurobiology of Cognition. A Cognition Special Issue*, MIT Press.

[Elman, 1988]: Elman, J.L. (1988). Finding Structure in Time. *Cognitive Science*, Vol. 14, No. 2, pp. 179-212.

[Elman, 1988]: Elman, J.L. (1988). Finding Structure in Time. *Technical Report* CRL 8801, Center for the Research of Language, UCSD, La Jolla, CA.

[Elman, 1991a]: Elman, J.L. (1991). Distributed Representations, Simple Recurrent Networks and Grammatical Structure. *Machine Learning*. Special Issue on Connectionist Approaches to Language Learning, Vol. 7, No. 2/3, pp. 195-226.

[Elman, 1991b]: Elman, J.L. (1991). Incremental Learning, or The Importance of Starting Small. *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, August 7-10, Chicago, IL, pp. 443-448.

[Erdi, 1990]: Erdi, P. (1990). Self-Organization in the Nervous System: Network Structure and Stability. *Technical Report* TR-KFKI-1990-60/H, Hungarian Academy of Sciences, Central Research Institute for Physics, Budapest, Hungary.

[Fanty, 1985]: Fanty, M. (1985). Context-Free Parsing in Connectionist Networks. *Technical Report* TR 174, Computer Science, University of Rochester, NY.

[Feldman et al., 1982]: Feldman, J.A. and Ballard, D.H. (1982). Connectionistic Models and their Properties. *Cognitive Science*, Vol. 6, pp. 205-254.

[Feldman et al., 1990]: Feldman, J.A., Lakoff, G., Stolcke A. and Weber, S.H. (1990). Miniature Language Acquisition: A Touchstone for Language Acquisition. *Technical Report* TR-90-009, ICSI Berkeley, CA.

[Feldman, 1981]: Feldman, J.A. (1981). A Connectionist Model of Visual Memory. In: *Parallel Models of Associative Memory* (G.E. Hinton and J.A. Anderson, Eds.), pp. 65-97. Lawrence Erlbaum.

[Feldman, 1989]: Feldman, J.A. (1989). What Lies Ahead. *Byte*, January 1989, pp. 348-350.

[Fillmore, 1968]: Fillmore, C.J. (1968). The Case for Case. In: *Universals in Linguisic Theory*, (E. Bach and R.T. Harms, Eds.).

[Fodor et al., 1988]: Fodor, J.A. and Pylyshyn, Z.W. (1988). Connectionism and Cognitive Architecture: A Critical Analysis. In: *Connections and Symbols* (J.A. Fodor and Z.W. Pylyshyn, Eds.), MIT Press.

[Forney, 1973]: Forney Jr., G.D. (1973). The Viterbi Algorithm. *Proceedings of the IEEE*, Vol. LXI, pp. 268-278.

[Fritzke et al., 1991]: Fritzke, B. and Wilke, P. (1991). FLEXMAP - A Neural Network For The Traveling Salesman Problem With Linear Time and Space Complexity. *Proceedings of the International Joint Conference on Neural Networks*, November 18-21, Singapore, pp. 929-934.

[Fritzke, 1991a]: Fritzke, B. (1991). Let it Grow - Self-Organizing Feature Maps with Problem Dependent Cell Structure. In: *Artificial Neural Networks* (T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, Eds.), Vol. 1, pp. 403-408. Elsevier Science Publishers.

[Fritzke, 1991b]: Fritzke, B. (1991). Unsupervised Clustering with growing Cell Structures. *Proceedings of the International Joint Conference on Neural Networks*. July. 8-12, Seattle, WA, Vol. 2, pp. 531-536.

[Fritzke, 1992a]: Fritzke, B. (1992). Wachsende Zellstrukturen - Ein Selbstorganisierendes Neuronales Netzwerkmodell (In German). *Ph.D. Thesis*, Universität Erlangen-Nürnberg.

[Fritzke, 1992b]: Fritzke, B. (1992). Growing Cell Structures - a Self-organizing Network in *k* Dimensions. In: *Artificial Neural Networks 2* (I. Aleksander and J. Taylor, Eds.), Vol. 2, pp. 1051-1056. Elsevier Science Publishers.

[Fu, 1977]: Fu, K.S. (Editor) (1977). *Syntactic Pattern Recognition, Applications*. Springer-Verlag.

[Gallant, 1988]: Gallant, S.I. (1988). Connectionist Expert Systems. *Communications of the ACM*, Vol. 31, No. 2, pp. 152-169.

[Garside et al., 1987]: Garside, R., Leech, G. and Sampson, G. (1987). *The Computational Analysis of English: a Corpus-based Approach*. Longman Group.

[Gellert et al., 1975]: Gellert, W., Kustner, H., Hellwich, M. and Kastner, H. (1975). *The VNR Concise Encyclopedia of Mathematics*. Van Nostrand Reinhold.

[Gersho et al., 1990a]: Gersho, M. and Reiter, R. (1990). Information Retrieval using Self-Organizing and Heteroassociative Supervised Neural Network. *Proceedings of the International Neural Network Conference*, July 9-13, Paris, France. Vol. 1, pp. 361-364.

[Gersho et al., 1990b]: Gersho, M. and Reiter, R. (1990). Information Retrieval using a Hybrid Multi-Layer Neural Network. *Proceedings of the International Joint Conference on Neural Networks*, June 17-21, San Diego, CA. Vol. 2, pp. 111-117.

[Gherrity, 1989]: Gherrity, M.A. (1989). A Learning Algorithm for Analog, Fully Recurrent Neural Nets. *Proceedings of the International Joint Conference on Neural Networks*, June 18-22, Washington DC, Vol 1, pp. 643-644.

[Gigley, 1983]: Gigley, H.M. (1983). HOPE -- AI and the Dynamic Process of Language Behaviour. *Cognition and Brain Theory*, Vol. 6, No. 1.

[Gigley, 1985]: Gigley, H.M. (1985). Computational Neurolinguistics-What is it all about? *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pp. 260-266.

[Giles et al., 1990]: Giles, C.L., Sun, G.Z., Chen, H.H., Lee, Y.C. and Chen, D. (1990). Higher Order Recurrent Networks and Grammatical Inference. In: *Advances in Neural Information Processing Systems* (D.S. Touretsky, Editor), Vol. 2, pp. 380-387. Morgan Kaufmann.

[Giles et al., 1991]: Giles, C.L., Miller, C.B., Chen, D., Chen, H.H., Sun, G.Z. and Lee, Y.C. (1991). Learning and Extracting Finite State Aytomata with Second-Order Recurrent Neural Networks. *Neural Computation*, Vol. 4, No. 3, pp. 393-405.

[Goldberg et al., 1988]: Goldberg, D.E. and Holland, J.H. (Editors) (1988). *Machine Learning* (Special Issue on Genetic Algorithms). Vol. 3, No. 2-3.

[Goldberg, 1989]: Goldberg, D.E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley Publishing Company.

[Golgi, 1906]: Golgi, C. (1906). The Neuron Doctrine. Theory and Facts. In: *Nobel Lectures: Physiology and Medicine*, 1901-1921, pp. 189-217. Elsevier Science Publishers.

[Graubard, 1988]: Graubard, S.R. (Editor) (1988). *The Artificial Intelligence Debate. False Starts, Real Foundations*. MIT Press.

[Grossberg, 1976]: Grossberg, S. (1976). Adaptive Pattern Classification and Universal Recoding: I. Parallel Development and Coding of Neural Feature Detectors. *Biological Cybernetics*, Vol. 23, pp. 121-134.

[Grossberg, 1980]: Grossberg, S. (1980). How Does a Brain Build a Cognitive Code? *Psychological Review*, Vol. 87, pp. 1-51.

[Grossberg, 1988]: Grossberg, S. (1988). Nonlinear Neural Nets. *Neural Networks*, Vol. 1, Nr. 1, pp. 17-61.

[Gutknecht et al., 1990]: Gutknecht, M. and Pfeifer, R. (1990). An Approach to Integrating Expert Systems with Connectionist Networks. *AI Communications*, Vol. 3, No. 3, pp. 116-127.

[Hanson et al., 1974]: Hanson, A.R., Riseman, E.M. (1974). A Contextual Postprocessing System for Error Correction Using Binary n-Grams. *IEEE Transactions on Computers*, May 1974, pp. 480-493, pp. 141-154.

[Hanson et al., 1987]: Hanson, S.J. and Kegl, J. (1987). PARSNIP: A Connectionist Network that Learns Natural Language Grammar from Exposure to Natural Language Sentences. *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, July 16-18. Seattle, WA, Seattle.

[Harris, 1974]: Harris, L.R. (1974). Natural Language Acquisition by Robot. *Technical Report* TR74-1, Department of Mathematics, Darthmouth College, NH.

[Harrison, 1971]: Harrison, M.C. (1971). Implementation of the Substring Test by Hashing. *Communications of the ACM*, Vol. 14, No. 12, pp. 777-779.

[Hayes-Roth, 1985]: Hayes-Roth, B. (1985). A Blackboard System for Control. *Artificial Intelligence*, Vol. 26, pp. 251-321.

[Hays, 1962]: Hays, D.G. (1962). Automatic Language-Data Processing. In: *Computer Applications in the Behavioral Sciences* (H. Borho, Editor), Prentice-Hall.

[Hebb, 1949]: Hebb, D.O. (1949). *The Organization of Behavior: A Neuropsychological Theory.* John Wiley & Sons.

[Hecht-Nielsen, 1987]: Hecht-Nielsen, R. (1987). Kolmogorov's Mapping Neural Networks Existance Therorem. *Proceedings of the IEEE International Conference on Neural Networks*, June 21-24, San Diego, CA. Vol. 2, pp. 332-339.

[Hecht-Nielsen, 1987]: Hecht-Nielsen, R. (1987). Counterpropagation Networks. *Applied Optics*, Vol. 26, December 1. 1987, pp. 4979-4984.

[Hecht-Nielsen, 1990]: Hecht-Nielsen, R. (1990). *Neurocomputing.* Addison Wesley Publishing Company.

[Hemani, et al., 1990]: Hemani, A. and Postula, A. (1990). Scheduling by Self-Organization. *Proceedings of the International Joint Conference on Neural Networks*, January 15-19, Washington DC, Vol. 2, pp. 543-546.

[Hendler, 1989]: Hendler, J.A. (1989). Spreading Activation over Distributed Microfeatures. In: *Advances in Neural Information Processing Systems* (D.S. Touretsky, Editor), Vol. 1, pp. 553-559. Morgan Kaufmann.

[Henseler et al., 1987]: Henseler, J., Scholtes, J.C. and Verhoest, C.R.J. (1987). The Design of a Parallel Knowledge-Based Optical-Character Recognition System. *Master Science Theses*, Delft University of Technology.

[Hingston et al., 1990]: Hingston, P. and Wilkinson, R. (1990). Document Retrieval Using a Neural Network. *Technical Report*, Department of Computer Science, RMIT at the University of Melbourne, 1990.

[Hinton et al., 1989]: Hinton, G.E. and Anderson, J.A. (Eds.) (1989). *Parallel Models of Associative Memory*, Updated Edition. Lawrence Erlbaum.

[Hinton, 1981]: Hinton, G.E. (1981). Implementing Semantic Networks in Parallel Hardware. In: *Parallel Models of Associative Memory*, (G.E. Hinton and J.A. Anderson, Eds.), Lawrence Erlbaum.

[Holland, 1975]: Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

[Honavar et al., 1989]: Honavar, V. and Uhr, L. (1989). Generation, Local Receptive Fields and Global Convergence Improve Perceptual Learning in Connectionist Networks. *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Vol. 1, pp. 180-185.

[Honavar et al., 1990]: Honavar, V. and Uhr, L. (1990). Symbol Processing Systems, Connectionist Networks, and Generalized Connectionist Networks. *Technical Report* TR#90-24, Department of Computer Science, Iowa State University, Ames, Iowa.

[Honkela et al., 1991]: Honkela, T. and Vepsalainen, A.M. (1991). Interpreting Imprecise Expressions: Experiments with Kohonen's Self-Organizing Maps and Associative Memory. In: *Artificial Neural Networks* (T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, Eds.), Vol. 1, pp. 897-902. Elsevier Science Publishers.

[Hopfield, 1982]: Hopfield, J.J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Science*, Vol. 79, pp. 2554-2558.

[Hrycej, 1989]: Hrycej, T. (1989). Unsupervised Learning by Backward Inhibition. *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pp. 170-175.

[Hrycej, 1990]: Hrycej, T. (1990). Self-Organization by Delta Rule. *Proceedings of the International Joint Conference on Neural Networks*, June 17-21, San Diego, CA, Vol. 2, pp. 307-312.

[Huang et al., 1987]: Huang, W.Y. and Lippmann, R.P. (1987). Comparision Between Neural Nets and Conventional Classifiers. *Proceedings IEEE International Conference on Neural Networks*, June 21-24, San Diego, Vol. 4, pp. 485-494.

[Hull et al., 1982]: Hull, J.J., Srihari, S.N. (1982). Experiments in Text Recognition with Binary n-Grams and Viterbi Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 520-530.

[Humphrey, 1989]: Humphrey, T.L. and Zhou, F. (1989). Period Disambiguation Using a Neural Network. *Proceedings of the International Joint Conference on Neural Networks*, June 18-22, Washington DC, Vol. 2, pp. 606.

[Ichiki et al., 1991]: Ichiki, H., Hagiwara, M., Nakagawa, N. (1991). Self-Organizing Multi-Layer Semantic Maps. *Proceedings of the International Joint Conference on Neural Networks*. July. 8-12, 1991. Seattle, WA, Vol. 1, pp. 357-360.

[Jagota, 1990]: Jagota, A. (1990). Applying a Hopfield-style Network to Degraded Printed Text Restoration. *Proceedings of the 3rd Conference on Neural Networks and Parallel Distributed Processing*, Indiana-Purdue University.

[Jagota, 1990]: Jagota, A. (1990). Applying a Hopfield-Style Network to Degraded Text Recognition. *Proceedings of the International Joint Conference on Neural Networks* 1990, Vol. 2, pp. 607-610.

[Jagota, 1991]: Jagota, A. (1991). Novelty Detection on a Very Large Number of Memories in a Hopfield-style Network. *Proceedings of the International Joint Conference on Neural Networks*, July 8-12, 1991. Seattle, WA, Vol. 2, pp. 905.

[Jagota, et al., 1990]: Jagota, A. and Hung, Y.-S. (1990). A Neural Lexicon in a Hopfield-style Network. *Proceedings of the International Joint Conference on Neural Networks*, January 15-19, Washington DC, Vol. 2, pp. 607-610.

[Jain et al., 1990]: Jain, A.N. and Waibel, A.H. (1990). Incremental Parsing by Modular Recurrent Connectionist Networks. In: *Advances in Neural Information Processing Systems* (D.S. Touretsky, Editor), Vol. 2, pp. 364-371. Morgan Kaufmann.

[James, 1890]: James, W. (1890). *The Principles of Psychology*. Holt, Reprint Dover.

[Jardine et al., 1971]: Jardine, N., Rijsbergen van, C.J. (1971). The Use of Hierarchic Clustering in Information Retrieval. *Information Storage and Retrieval*, Vol. 7, No. 5, pp. 217-240.

[Jelinek et al., 1989]: Jelinek, F., Fusijaki, T., Cocke, J., Black, E. and Nishino, T. (1989). A Probabilistic Parsing Method for Sentence Disambiguation. *CMU International Parsing Workshop*, 1989, pp. 85-94.

[Jelinek et al., 1991a]: Jelinek, F. (1991). Up from Trigrams: The Struggle for Improved Language Models. *Technical Report*, IBM, T.J. Watson Research Center, New York.

[Jelinek et al., 1991b]: Jelinek, F. (1991). Self-Organized Language Modelling for Speech Recognition. *Technical Report*, IBM, T.J. Watson Research Center, New York.

[Jelinek et al., 1991c]: Jelinek, F. and Lafferty, J.D. (1991). Computation of the Probability of Initial Substring Generation by Stochastic Context Free Grammars. *Technical Report*, IBM, T.J. Watson Research Center, New York.

[Jodouin, 1990]: Jodouin, J.F. and Memmi, D. (1990). The Waltz and Pollack Model of Natural Language Processing: Critiques and Extensions. *Proceedings of the International Neural Network Conference*, July 9-13, Paris, France, pp. 541-544.

[Jordan, 1986]: Jordan, M.I. (1986). Serial Order: A Parallel Distributed Processing Approach. Institute for Cognitive Science, *Technical Report* #8604, UCSD, La Jolla, CA.

[Jordan, 1986]: Jordan, M.I. (1986). Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. *Proceedings of the Cognitive Science Society*, Amherst, MA, pp. 531-546.

[Jordan, 1988]: Jordan, M.I. (1988). Supervised Learning and Systems with Excess Degrees of Freedom. *Technical Report*, COINS TR 88-27, MIT Press.

[Jordan, 1988]: Jordan, M.I. (1988). Supervised Learning and Systems with Excess Degree of Freedom. In: *Proceedings of the Connectionist Models Summer School* (D.S. Touretzky, G.E. Hinton and T.J. Sejnowski, Eds.), Carnegie Mellon University, PA, pp. 62-75. Morgan Kaufmann.

[Jung et al., 1991]: Jung, G.S. and Raghavan, V.V. (1991). Connectionist Learning in Constructing Thesaurus-Like Knowledge Structure. *AAAI Symposium on Text-Based Inteligent Systems*, Stanford University, Palo Alto, CA.

[Kamimura, 1990a]: Kamimura, R. (1990). Experimental Analysis of Performance of Temporal Supervised Learning Algorithm, Applied to a Long and Complex Sequence. *Proceedings of the International Neural Network Conference*, July 9-13, Paris, France, pp. 753-756.

[Kamimura, 1990b]: Kamimura, R. (1990). Application of Temporal Supervised Learning Algoritm to Generation of Natural Language. in: *Proceedings of the IEEE International Joint Conference on Neural Networks*, June 17-21, San Diego, CA. Vol. 1, pp. 201-207.

[Kandel et al.,1985]: Kandel, E.R. and Schwartz, J.H. (1985). *Principles of Neural Science (Second Edition)*. Elsevier Science Publishers.

[Kangas et al., 1990]: Kangas, J.A., Kohonen, T.K. and Laaksonen, J.T. (1990). Variants on Self-Organizing Maps. *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, pp. 93-99.

[Kangas, 1990]: Kangas, J.A. (1990). Time-Delayed Self-Organizing Maps. *Proceedings of the International Joint Conference on Neural Networks*, June 17-21, San Diego, CA. Vol. 2, pp. 331-336.

[Kangas, 1992]: Kangas, J.A. (1992). Temporal Knowledge in Locations of Activations in a Self-Organizing Map. In: *Artificial Neural Networks 2* (I. Aleksander and J. Taylor, Eds.), Vol. 1, pp. 117-120. Elsevier Science Publishers.

[Kehagias, 1990]: Kehagias, A. (1990). Reproducing Infinite Boolean Sequences: An Application of Hidden Markov Models to Connectionist Learning. *Proceedings of the International Joint Conference on Neural Networks*, January 15-19, Washington DC, Vol. 1, pp. 281-284.

[Kelly, 1991]: Kelly, M. (1991). Self-Organizing Map Training Using Dynamic K-D Trees. In: *Artificial Neural Networks* (T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, Eds.), Vol. 2, pp. 1041-1044. Elsevier Science Publishers.

[Kimbrell, 1988]: Kimbrell, R.E. (1988). Searching for Text? Send an N-Gram! *Byte*, May 1988, pp. 297-312.

[Knuth et al., 1977]: Knuth, D.E., Morris, J.H., Pratt, V.R. (1977). Fast Pattern Matching In Strings. *SIAM Journal of Computing*, Vol. 6, No. 2, pp. 323-350.

[Kodratoff et al., 1990]: Kodratoff, Y. and Michalski, R.S. (Eds.) (1990). *Machine Learning, An Artificial Intelligence Approach. Volume 3*. Morgan Kaufmann.

[Kohonen et al., 1981]: Kohonen, T., Oja, E. and Lehtio, P. (1981). Storage and Processing of Information in Distributed Associative Memory Systems. In: *Parallel Models of Associative Memory* (G.E. Hinton & J.A. Anderson, Eds.), Lawrence Erlbaum.

[Kohonen, 1972]: Kohonen, T. (1972). Correlation Matrix Memories. *IEEE Transactions on Computers*, Vol. C-21, pp. 353-359.

[Kohonen, 1977]: Kohonen, T. (1977). *Associative Memory: A Systems-Theoretical Approach.* Springer-Verlag.

[Kohonen, 1982a]: Kohonen, T. (1982). Analysis of a Simple Self-Organizing Process. *Biological Cybernetics*, Vol. 44, pp. 135-140.

[Kohonen, 1982b]: Kohonen, T. (1982). Clustering, Taxonomy, and Topological Maps of Patterns. *Proceedings of the 6th International Conference on Pattern Recognition*, pp. 114-128.

[Kohonen, 1982c]: Kohonen, T. (1982). Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, Vol. 43, pp. 59-69.

[Kohonen, 1984]: Kohonen, T. (1984). *Self-Organization and Associative Memory.* Springer-Verlag.

[Kohonen, 1987]: Kohonen, T. (1987). *Content Addressable Memories, 2nd Edition.* Springer-Verlag.

[Kohonen, 1988]: Kohonen, T. (1988). An Introduction to Neural Computing. *Neural Networks*, Vol. 1, nr. 1, pp. 3-16.

[Kohonen, 1990a]: Kohonen, T. (1990). Some Practical Aspects of the Self-Organizing Maps. *Proceedings of the International Joint Conference on Neural Networks*, January 15-19, Washington DC, Vol. 2, pp. 253-256.

[Kohonen, 1990b]: Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE*, Vol. 78, pp. 1464-1480.

[Kohonen, 1991]: Kohonen, T. (1991). The Hypermap Architecture. In: *Artificial Neural Networks* (T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, Eds.), Vol. 2, pp. 1357-1362. Elsevier Science Publishers.

[Koikkalainen et al., 1990]: Koikkalainen, P. and Oja, E. (1990). Self-Organizing Hierarchical Feature Maps. *Proceedings of the International Joint Conference on Neural Networks*, June 17-21, 1990, San Diego, CA. Vol. 2, pp. 279-284.

[Kraaijveld et al., 1991]: Kraaijveld, M., Duin, R. (1991). Generalization Capabilities of Minimal Kernal-based Networks. *Proceedings of the International Joint Conference on Neural Networks*. July. 8-12, 1991. Seattle, WA. Vol. 1, pp. 843-848.

[Kuhn et al., 1990]: Kuhn, G.M., Watrous, R.L. and Ladendorf, B. (1990). Connected Recognition with a Recurrent Network. *Speech Communication*, Vol. 9 (1), pp. 41-48.

[Kuhnel et al., 1990]: Kuhnel, H., Tavan, T. (1990). The Anti-Hebb Rule Derived from Information Theory. In: *Parallel Processing in Neural Systems and Computers* (R. Eckmiller, G. Hartmann and G. Hauske, Eds), pp. 187-190. Elsevier Science Publishers.

[Kukich, 1988]: Kukich, K. (1988). Back Propagation Topologies for Sequence Generation. *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, Vol. 1, pp. 301-308.

[Kwok, 1989]: Kwok, K.L. (1989). A Neural Network for Probabilistic Information Retrieval. *Proceedings of the 12th ACM-SIGIR Conference on Research & Development in Information Retrieval.* June 11-20, 1989. Cambridge, MA, pp. 21-30.

[Kwok, 1990]: Kwok, K.L. (1990). Application of Neural Network to Information Retrieval. *Proceedings of the International Joint Conference on Neural Networks*, January 15-19. Washington DC. Vol. 2, pp. 623-626.

[Kwok, 1991]: Kwok, K.L. (1991). Query Modofication and expanding in a Network with Adaptive Architecture. *Proceedings of the 14th Annual International ACM/SIGIR Conference on Research & Development in Information Retrieval*, Chicago, IL, pp. 192-201.

[Lakoff, 1988]: Lakoff, G. (1988). A Suggestion for a Linguists with Connectionist Foundations. In: *Proceedings of the Connectionist Models Summer School* (D.S. Touretsky, G.E. Hinton and T.J. Sejnowski, Eds.), Carnegie Mellon University, PA, pp. 301-314. Morgan Kaufmann.

[Lancaster, 1979]: Lancaster, F.W. (1979). *Information Retrieval Systems: Characteristics, Testing and Evaluation (Second Edition)*. John Wiley & Sons.

[Lange et al., 1989]: Lange, T.E. and Dyer, M.G. (1989). Dynamic, Non-Local Role Bindings and Inferencing in a Localist Network for Natural Language Understanding. In: *Advances in Neural Information Processing Systems* (D.S. Touretsky, Editor), Vol. 1, pp. 545-552. Morgan Kaufmann.

[Larson, 1988]: Larson, P.A. (1988). Dynamic Hash Tables. *Communications of the ACM*, Vol. 31, nr. 4, pp. 446-457.

[Le Cun, 1986]: Le Cun, Y. (1986). Learning Process in an Asymmetric Threshold Network. In: *Disordered Systems and Biological Organizations* (E.Bienenstock, F. Fogelman Souli, and G. Weisbuch, Eds.), Springer-Verlag.

[Lelu, 1989]: Lelu, A. (1989). Local Component Analysis: A Neural Model for Information Retrieval. *Proceedings of the International Joint Conference on Neural Networks*, June 18-22, Washington DC, Vol. 2, pp. 43-47.

[Lelu, 1991]: Lelu, A. (1991). From Data Analysis to Neural Networks: New Prospects for Efficient Browsing through Databases. *Journal of Information Science*, Vol. 17, pp. 1-12. Elsevier Science Publishers.

[Lieberman, 1991]: Lieberman, M.Y. (1991). The Trend Towards Statistical Models in Natural Language Processing. In: *Natural Language Processing and Speech Processing Symposium*, pp. 1-7. Springer-Verlag.

[Lin et al., 1991]: Lin, X., Soergel, D., and Marchionini, G. (1991). A Self-organizing Semantic Map for Information Retrieval. *Proceedings of the 14th Annual International ACM-SIGIR Conference on Research & Development in Information Retrieval*, Chicago, IL. pp. 262-269.

[Linsker, 1987]: Linsker, R. (1987). Towards an Organizing Principle for a Layered Perceptual Network. In: *Neural Information Processing Systems* (D.Z. Anderson, Editor), pp. 485-494. Amererican Institute of Physics.

[Linsker, 1988]: Linsker, R. (1988). Self-Organization in a Perceptual Network. *IEEE Computer, Special Issue on Artificial Neural Systems*, Vol. 21, nr. 3, pp. 105-117.

[Linsker, 1989a]: Linsker, R. (1989). An Application of the Principle of Maximum Information Preservation to Linear Systems. In: *Advances in Neural Information Processing Systems* (D.S. Touretzky, Editor), Vol. 1, pp. 186-194. Morgan Kaufmann.

[Linsker, 1989b]: Linsker, R. (1989). How to Generate Ordered Maps by Maximizing the Mutual Information between Input and Output Signals. *Neural Computation*, Vol. 1, pp. 396-405.

[Linsker, 1990]: Linsker, R. (1990). Perceptual Neural Organization: Some Approaches Based on Network Models and Information Theory. *Annual Review of Neuroscience*, Vol. 13, pp. 257-281.

[Lippmann, 1987]: Lippmann, R.P. (1987). An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, April 1987, pp. 4-22.

[Liu et al., 1990]: Liu, Y.D., Sun, G.Z., Chen, H.H. and Lee, Y.C. (1990). Grammatical Inference and Neural Network State Machines. *Proceedings of the International Joint Conference on Neural Networks*, January 15-19, Washington DC. Vol. 1, pp. 285-288.

[Malsberg, 1973]: Malsberg, Chr. von der (1973). Self-Organization of Orientation Sensitive Cells in the Striate Cortex. *Kybernetik*, Vol. 14, pp. 85-100.

[Marcus, 1980]: Marcus, M.P. (1980). *A Theory of Syntactic Recognition for Natural Language*. MIT Press.

[Martinetz et al., 1991]: Martinetz, T. and Schulten, K. (1991). A "Neural-Gas" Network Learns Topologies. In: *Artificial Neural Networks* (T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, Eds.), Vol. 1, pp. 397-402. Elsevier Science Publishers.

[Mauldin, 1991]: Mauldin, M.L. (1991). *Conceptual Information Retrieval: A Case Study in Adaptive Partial Parsing*. Kluwer Academic Publishers.

[McClelland et al., 1981]: McClelland, J.L. and Rumelhart, D.E. (1981). An Interactive Activation Model of Context Effects in Letter Perception. *Psychological Review*, Vol. 88, pp. 375-407.

[McClelland et al., 1986a]: McClelland, J.L. and Rumelhart, D.E. (Eds.) (1987). *Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models.* MIT Press.

[McClelland et al., 1986b]: McClelland, J.L. and Rumelhart, D.E. (1986). *Explorations in Parallel Distributed Processing. A Handbook of Models, Programs, and Exercises.* MIT Press.

[McClelland et al., 1986c]: McClelland, J.L. and Elman, J.L. (1986). The TRACE Model of Speech Perception. *Cognitive Psychology*, Vol. 18, pp. 1-86.

[McClelland et al., 1986d]: McClelland, J.L. and Kawamoto, A.H. (1986). Mechanisms of Sentence Processing: Assigning Roles to Constituents of Sentences. In: *Parallel Distributed Processing Vol. 2* (J.L. McClelland and D.E. Rumelhart, Eds.), MIT Press.

[McCulloch et al., 1943]: McCulloch, W.S. and Pitts, W.A. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematics and Biophysics*, Vol. 5, pp. 115-133.

[McCulloch, 1990]: McCulloch, N. (1990). Recurrent Networks for Learning Stochastic Sequences. *Proceedings of the International Neural Network Conference*, July 9-13, Paris, France, pp. 1032-1035.

[McIlroy, 1982]: McIlroy, M.D. (1982). Development of a Spelling List. *IEEE Transactions on Communications*, Vol. COM-30, pp. 91-99.

[Michalski et al., 1986a]: Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (Eds.) (1986). *Machine Learning, An Artificial Intelligence Approach. Volume 1.* Morgan Kaufmann.

[Michalski et al., 1986b]: Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (Eds.) (1986). *Machine Learning, An Artificial Intelligence Approach. Volume 2.* Morgan Kaufmann.

[Miikkulainen et al., 1988a]: Miikkulainen, R. and Dyer, M.G. (1988). Encoding Input/ Output Representations in Connectionist Cognitive Systems. In: *Proceedings of the Connectionist Models Summer School* (D.S. Touretzky, G.E. Hinton, and T.J. Sejnowski, Eds.), Carnegie Mellon University, PA, pp. 347-356. Morgan Kaufmann.

[Miikkulainen et al., 1988b]: Miikkulainen, R. and Dyer, M.G. (1988). Forming Global Representations with Extended Back Propagation. *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA.

[Miikkulainen, 1990a]: Miikkulainen, R. (1990). A Distributed Feature Map Model of the Lexicon. *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, July 25-28, Cambridge, MA. 1990, pp. 447-454.

[Miikkulainen, 1990b]: Miikkulainen, R. (1990). Script Recognition with Hierarchical Feature Maps. *Connection Science*, Vol. 2, No. 1&2, pp. 83-102.

[Miller, 1956]: Miller, A.M. (1956). The Magical Number Sevewn, Plus or Minus Two: Sone Limits to our Capacity for Processing Information. *Psychological Review*, Vol. 63, pp. 81-97.

[Minsky et al., 1969/1988]: Minsky, M.L. and Papert, S. (1969). *Perceptrons (2nd Edition)*. MIT Press.

[Mitzman et al., 1990]: Mitzman, D. and Giovannini, R. (1990). ActivityNets: A Neural Classifier of Natural Language Descriptions of Economic Activities. *Proceedings of the International Workshop on Neural Nets for Statistics and Economic Data*, December 10-11, 1991. Dublin, Ireland.

[Mozer, 1984]: Mozer, M.C. (1984). Inductive Information Retrieval Using Parallel Distributed Computation. *Technical Report* TR-ICS 8406, UCSD, La Jolla, CA.

[Mozer, 1991]: Mozer, M.C. (1991). Discovering Discrete Distributed Representations. In: *Advances in Neural Information Processing Systems 3* (R.P. Lippmann, J.E. Moody and D.S. Touretzky, Eds.), Vol. 3, pp. 627-634. Morgan Kaufmann.

[Neuhoff, 1975]: Neuhoff, D.L. (1975). The Viterbi Algorithm As an Aid in Text Recognition. *IEEE Transactions on Information Theory*, Vol. IT-XXI, pp. 222-226.

[Nijholt, 1990]: Nijholt, A. (1990). Meta-Parsing in Neural Networks. *Memoranda Informatica* 90-08, Universiteit van Twente, The Netherlands.

[Oddy et al., 1991]: Oddy, R.N. and Balakrishnan, B. (1991). PThomas: an Adaptive Information Retrieval System on the Connection Machine. *Information Processing and Management.*

[Oja et al., 1988]: Oja, E. and Kohonen, T. (1988). The Subspace Learning Algorithm as a Formalism for Pattern Recognition and Neural Networks. *Proceedings of the 2nd International Conference on Neural Networks*, San Diego, CA. Vol. 1, pp. 277-284.

[Oja et al., 1991]: Oja, E., Ogawa, H. and Wangviwattana, J. (1991). Learning in Nonlinear Constrained Hebbian Networks. In: *Artificial Neural Networks* (T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, Eds.), Vol. 1, pp. 385-390. Elsevier Science Publishers.

[Ojemann, 1983]: Ojemann, G.A. (1983). Brain Organization for Language from the Perspective of Electrical Stimulating Mapping. *Behav. Brain Science*, pp. 189-230.

[Palakal et al., 1991]: Palakal, M. and Thai, X. (1991). Computational Model of a Biologically Plausible Cognitive map. *Proceedings of the International Joint Conference on Neural Networks*. July. 8-12, 1991. Seattle, WA. Vol. 2, pp. A-885.

[Parker, 1985]: Parker, D. (1985). Learning Logic. *Technical Report* TR-87, Center for Computational Research in Economics and Management Science, MIT Press.

[Pavlidis, 1977]: Pavlidis, T. (1977). *Structural Pattern Recognition*. Springer-Verlag.

[Penfield, 1950]: Penfield, W. and Rasmussen, T. (1950). *The Cerebral Cortex of Man: A Clinical Study of Localization of Function*. Macmillan.

[Personnaz et al., 1986]: Personnaz, L., Guyon, I. and Dreyfus, G. (1986). Neural Network Design for Efficient Information Retrieval. In: *Disordered Systems and Biological Organization* (E. Bienenstock et al., Eds.), Springer-Verlag.

[Pineda, 1987]: Pineda, F.J. (1987). Generalization of Back Propagation to Recurrent Neural Networks. *Psychological Review Letters*, Vol. 59, nr. 19, pp. 2229-2232.

[Pineda, 1988]: Pineda, F.J. (1988). Generalization of Back Propagation to Recurrent and Higher-Order Neural Networks. In: *Neural Information Processing Systems* (D.Z. Anderson, Editor), pp. 602-611. American Institute of Physics.

[Pineda, 1989]: Pineda, F.J. (1989). Recurrent Backpropagation and the Dynamical Approach to Adaptive Neural Computation. *Neural Computation*, Vol. 1, No. 2, pp. 161-172.

[Pinker et al., 1988]: Pinker, S. and Prince, A. (1988). On Language and Connectionism: Analysis of a Parallel Distributed Processing Model of Language Acquisition. *Cognition*, Vol. 28, pp. 73-193.

[Pinker et al., 1988]: Pinker, S. and Prince, A. (1988). On Language and Connectionism: Analysis of a parallel distributed processing model of language acquisition. In: *Connections and Symbols* (J.A. Fodor, and Z.W. Pylyshyn), Elsevier Science Publishers.

[Pinker, 1989]: Pinker, S. (1989). Language Acquisition. In: *Foundations of Cognitive Science* (M.I. Posner, Editor), pp. 359-400. MIT Press.

[Pollack, 1990]: Pollack, J.B. (1990). Recursive Distributed Representations. *Artificial Intelligence*, Vol. 46, pp. 77-105. Elsevier Science Publishers.

[Powers, 1983]: Powers, D.M.W. (1983). Neurolinguistics and Psycholinguistics as a Basis for Computer Acquisition of Natural Language. *SIGART*, Vol. 84, pp. 29-34.

[Rabiner et al., 1986]: Rabiner, L.R. and Juang, B.H. (1986). An Introduction to the Hidden Markov Models. *IEEE ASAP Magazine*, pp. 4-16.

[Rapp et al., 1991a]: Rapp, R., Wettler, M. (1991). A Connectionist Simulation of Human Word Associations. *Proceedings of the International Joint Conference on Neural Networks*. July. 8-12, 1991. Seattle, WA. Vol. 2, pp. A-946.

[Reeke et al., 1988]: Reeke, G.N. and Edelman, G.M. (1988). Real Brains and Artificial Intelligence. In: *The AI Debate. False Starts, Real Foundations* (R. Graubard, Editor), pp. 143-174. MIT Press.

[Regier, 1988]: Regier, T. (1988). Recognizing Image-Schemes Using Programmable Networks. In: *Proceedings of the Connectionist Models Summer School*, (D.S. Touretzky, G.E. Hinton, and T.J. Sejnowski, Eds.), Carnegie Mellon University, PA, pp. 315-324. Morgan Kaufmann.

[Rijsbergen, 1979]: Rijsbergen, C.J. van (1979). *Information Retrieval.* Butterworths, London.

[Ritter et al., 1986]: Ritter, H. and Schulten, K. (1986). On the Stationary State of Kohonen's Self-Organizing Sensory Mapping. *Biological Cybernetics*, Vol. 54, pp. 99-106.

[Ritter et al., 1988]: Ritter, H. and Schulten, K. (1988). Kohonen's Self-Organizing Maps: Exploring their Computational Capabilities. *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA. Vol. 1, pp. 109-116.

[Ritter et al., 1989a]: Ritter, H. and Schulten, K. (1989). Convergency Properties of Kohonen's Topology Conserving Maps: Fluctuations, Stability and Dimension Selection. *Biological Cybernetics*, Vol. 60, pp. 59-71.

[Ritter et al., 1989b]: Ritter, H. and Kohonen, T. (1989). Self-Organizing Semantical Maps. *Biological Cybernetics*, Vol. 61, pp. 241-254.

[Ritter et al., 1990]: Ritter, H. and Kohonen, T. (1990). Learning "Semantotopic Maps" from Context. *Proceedings of the International Joint Conference on Neural Networks*, January 15-19, Washington DC. Vol. 1, pp. 23-26.

[Ritter et al., 1992]: Ritter, H., Martinetz, T., Schulten, K. (1992). *Neural Computation and Self-organizing Maps, An Introduction.* Addison Wesley Publishing Company.

[Ritter, 1989]: Ritter, H. (1989). Combining Self-Organizing Maps. *Proceedings of the International Joint Conference on Neural Networks*, June 18-22, Washington DC. Vol. 2, pp. 499-502.

[Robbins et al., 1951]: Robbins, H. and Monro, S. (1951). A Stochastic Appriximation Method. *Annals of Math. Stat.*, Vol. 22, pp. 400-407.

[Rose et al., 1989]: Rose, D.E. and Belew, R.K. (1989). A case for Symbolic/Sub-Symbolic Hybrids. *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, August. 16-19, 1989. Ann Arbor, MI. pp. 844-851.

[Rose et al., 1991]: Rose, D.E. and Belew, R.K. (1991). A Connectionist and Symbolic Hybrid for Improving Legal Research. *International Journal of Man-Machine Studies.*

[Rose, 1990]: Rose, D.E. (1990). Appropriate Uses of Hybrid Systems. In: *Connectionist Models: Proceedings of the 1990 Summer School* (D.S. Touretzky, J.L. Elman, T.J. Sejnowski, G.E. Hinton), pp. 277-286. Morgan Kaufmann.

[Rose, 1991]: Rose, D.E. (1991). A Symbolic and Connectionist Approach to Legal Information Retrieval. *Ph.D. Thesis*, UCSD, La Jolla, CA.

[Rosenblatt, 1958]: Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, Vol. 65, pp. 386-408.

[Rosenblatt, 1962]: Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan.

[Rubner et al., 1990]: Rubner, J. and Schulten, K. (1990). Development of Feature Detectors by Self-Organization. *Biological Cybernetics*, Vol. 62, pp. 193-199.

[Rueckl, 1986]: Rueckl, J.G. (1986). A Distributed Connectionist Model of Letter and Word Identification. *Ph.D. Thesis*, University of Wisconsin.

[Rumelhart et al., 1982]: Rumelhart, D.E. and McClelland, J.L. (1982). An Interactive Activation Model of Context Effects in Letter Perception: Part 2. *Psychological Review*, Vol. 89, pp. 60-94.

[Rumelhart et al., 1985]: Rumelhart, D.E. and Zipser, D. (1985). Feature Discovery by Competetive Learning. *Cognitive Science*, Vol. 9, pp. 75-112.

[Rumelhart et al., 1986a]: Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986). Learning Internal Representations by Error Propagation. In: *Parallel Distributed Processing. Vol. 1.* (D.E. Rumelhart and J.L. McClelland, Eds.), pp. 318-362. MIT Press.

[Rumelhart et al., 1986b]: Rumelhart, D.E. and McClelland, J.L. (1986). On Learning the Past Tenses of English Verbs. In: *Parallel Distributed Processing*, (J.L. McClelland and D.E. Rumelhart, Eds.), Vol. 2. pp. 216-271. MIT Press.

[Rumelhart et al., 1986c]: Rumelhart, D.E. and McClelland, J.L. (Eds.) (1986). *Explorations in the Microstructure of Cognition. Volume 1: Foundations*. MIT Press.

[Rumelhart et al., 1986d]: Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986). Learning Representations by Back-Propagating Errors. *Nature*, pp. 533-536.

[Saito et al., 1988]: Saito, K. and Nakano, R. (1988). Medical Diagnostic Expert System Based on PDP Model. *Proceedings of the IEEE International Conference on Neural Networks*, 1988, San Diego, CA.

[Salton et al., 1968]: Salton, G., Wong, A., Yang, C.S. (1968). A Vector Space Model for Automatic Indexing. *Communications of the ACM*, Vol. 18, No. 11, pp. 613-620.

[Salton et al., 1973]: Salton, G., Yang, C.S. (1973). On the Specification of Term Values in Automatic Indexing. *Journal of Documentation*, Vol. 29, no. 4, pp. 351-372.

[Salton et al., 1983a]: Salton, G. and McGill, M.J. (Eds.) (1983). *Introduction to Modern Information Retrieval.* McGraw-Hill.

[Salton et al., 1983b]: Salton, G., Fox, E.A., Wu, H. (1983). Extended Boolean Information Retrieval. *Communications of the ACM*, Vol. 26, No. 11, pp. 189-195.

[Salton et al., 1985]: Salton, G., Fox, E.A. and Voorhees, E. (1985). Advanced Feedback Methods in Information Retrieval. *Journal of the American Society for Information Science*, Vol. 36, No. 3, pp. 200-210.

[Salton et al., 1987]: Salton, G. and Buckley, C. (1987). Term Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 1987, 24, pp. 513-523.

[Salton et al., 1988a]: Salton, G. and Buckley, C. (1988). On the Use of Spreading Activation Methods in Automatic Information Retrieval. *Proceedings of the 11th International ACM-SIGIR Conference on Research & Development in Information Retrieval*, June 13-15, Grenoble, France, pp. 147-160.

[Salton et al., 1988b]: Salton, G. and Buckley, C. (1988). Parallel Text Search Methods. *Communications of the ACM*, Vol. 31, No. 2, pp. 202-215.

[Salton et al., 1991]: Salton, G. and Buckley, C. (1991). Automatic Text Structuring and Retrieval: Experiments in Automatic Encyclopedia Searching. *Proceedings of the 14th*

Annual International ACM-SIGIR Conference on Research & Development in Information Retrieval, Chicago, IL. pp. 21-31.

[Salton, 1968]: Salton, G. (1968). *Automatic Information Organization and Retrieval.* McGraw-Hill Book Co., New York.

[Salton, 1971]: Salton, G. (Editor) (1971). *The Smart Retrieval System.* Prentice Hall.

[Salton, 1972]: Salton, G. (1972). Experiments in Automatic Thesaurus Construction for information Retrieval. *Information Processing & Management*, Vol. 71, pp. 115-123.

[Salton, 1980a]: Salton, G. (1980). Automatic Term Class Construction Using Relevance - A Summery of Work in Automatic Pseudoclassification. *Information Processing & Management.* 1980; 16(1), pp. 1-15.

[Salton, 1980b]: Salton, G. (1980). Automatic Information Retrieval. *IEEE Computer*, Vol. 13, No. 9, pp. 41-57.

[Salton, 1981]: Salton, G. (1981). A Blueprint for Automatic Indexing. *ACM SIGIR Forum*, 1981 Fall; 16(2), pp. 22-38.

[Salton, 1986]: Salton, G. (1986). Another Look at Automatic Text Retrieval. *Communications of the ACM*, Vol. 29, No. 7, pp. 648-656.

[Salton, 1989]: Salton, G. (1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley Publishing Company.

[Samarabandu, et al., 1990]: Samaranbandu, J.K. and Jakubowics, O.G. (1990). Principles of Sequential Feature Maps in Multi-level Problems. *Proceedings of the International Joint Conference on Neural Networks*, January 15-19, Washington DC. Vol. 2, pp. 683-686.

[Sammon, 1969]: Sammon, J.W. (1969). A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, Vol. 18, pp. 401-409.

[Santos, 1989]: Santos, E. (1989). A Massive Parallel Self-Tuning Context-Free Parser. In: *Advances in Neural Information Processing Systems* (D.S. Touretsky, Editor), Vol. 1, pp. 537-544. Morgan Kaufmann.

[Scha, 1990]: Scha, R. (1990). Taaltheorie en Taaltechnologie; Competence en Performance (In Dutch). In: *Computertoepassingen in de Neerlandistiek* (K. de Kort & G.L.J. Leerdam, Eds.), pp. 99-109. LVVN.

[Scha, 1992]: Scha, R. (1992). Virtuele Grammatica's en Creative Algoritmen (in Dutch). *Grammaltt*, Vol. 1 No. 1, pp. 57-77.

[Schalkoff, 1992]: Schalkoff, R. (1992). *Pattern Recognition.* John Wiley & Sons.

[Scholtes et al., 1992a]: Scholtes, J.C. and Bloembergen, S. (1992). The Design of a Neural Data-Oriented Parsing (DOP) Model. *Proceedings of the International Joint Conference on Neural Networks*, June 7-10, Baltimore, MD.

[Scholtes et al., 1992b]: Scholtes, J.C. and Bloembergen, S. (1992). Corpus Based Parsing with a Self-Organizing Neural Net. *Proceedings of the International Joint Conference on Neural Networks*, November 3-5, Beijing, P.R. China.

[Scholtes, 1990a]: Scholtes, J.C. (1990). Neurolinguistics. *Computational Linguistics Project*, CERVED S.p.A., Italy, 1990.

[Scholtes, 1990b]: Scholtes, J.C. (1990). Trends in Neurolinguistics. *Proceedings of the IEEE Symposium on Neural Networks*, June 21, Delft, Netherlands, pp. 69-86.

[Scholtes, 1991a]: Scholtes, J.C. (1991). Using Extended Kohonen-Feature Maps in a Language Acquisition Model. *Proceedings of the 2nd Australian Conference on Neural Nets*. February 2-4. Sydney, Australia, pp. 38-43.

[Scholtes, 1991b]: Scholtes, J.C. (1991). Learning Simple Semantics by Self-Organization. *Worknotes of the AAAI Spring Symposium Series on Machine Learning of Natural Language and Ontology.* March 26-29. Palo Alto, CA, pp. 146-151.

[Scholtes, 1991c]: Scholtes, J.C. (1991). Learning Simple Semantics by Self-Organization. *Worknotes of the AAAI Spring Symposium on Connectionist Natural Language Processing*. March 26-29. Palo Alto, CA, pp. 78-83.

[Scholtes, 1991d]: Scholtes, J.C. (1991). Recurrent Kohonen Self-Organization in Natural Language Processing. In: *Artificial Neural Networks* (T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, Eds.), pp. 1751-1754. Elsevier Science Publishers.

[Scholtes, 1991e]: Scholtes, J.C. (1991). Unsupervised Context Learning in Natural Language Processing. *Proceedings of the International Joint Conference on Neural Networks*. July 8-12. Seattle, WA, Vol. 1, pp. 107-112.

[Scholtes, 1991f]: Scholtes, J.C. (1991). Kohonen's Self-Organizing Map in Natural Language Processing. *Proceedings of the SNN Symposium.* May 1-2. Nijmegen, The Netherlands, pp. 64.

[Scholtes, 1991g]: Scholtes, J.C. (1991). Kohonen's Self-Organizing Map Applied Towards Natural Language Processing. *Proceedings of the CUNY 1991 Conference on Sentence Processing*. May 12-14. Rochester, NY, pp. 10.

[Scholtes, 1991h]: Scholtes, J.C. (1991). Kohonen Feature Maps in Natural Language Processing. *Technical Report* ITLI-CL-1, Department of Computational Linguistics. March 1991, University of Amsterdam.

[Scholtes, 1991i]: Scholtes, J.C. (1991). Self-Organized Language Learning. *The Annual Conference on Cybernetics: Its Evolution and Its Praxis*. July 17-21. Amherst, MA.

[Scholtes, 1991j]: Scholtes, J.C. (1991). Unsupervised Learning and the Information Retrieval Problem. *Proceedings of the International Joint Conference on Neural Networks*, November 18-21, Singapore., pp. 18-21,

[Scholtes, 1991k]: Scholtes, J.C. (1991). Kohonen Feature Maps and Full-Text Data Bases: A Case Study of the 1987 Pravda. *Proceedings of Informatiewetenschap 1991*. December 18. Nijmegen, The Netherlands, pp. 203-220. STINFON.

[Scholtes, 1991l]: Scholtes, J.C. (1991). Filtering the Pravda with a Self-Organizing Neural Net. *Working Notes of the Bellcore Workshop on High Performance Information Filtering*. November 5-7, Chester, NJ.

[Scholtes, 1991m]: Scholtes, J.C. (1991). Neural Nets and Their Relevance for Information Retrieval. *Technical Report* ITLI-CL-2, Department of Computational Linguistics. August 1991, University of Amsterdam.

[Scholtes, 1992a]: Scholtes, J.C. (1992). Neural Nets versus Statistics in Information Retrieval. A Case Study of the 1987 Pravda. *Proceedings of the SPIE Conference on Applications of Artificial Neural Networks III*, April 20-24, Orlando, FL.

[Scholtes, 1992b]: Scholtes, J.C. (1992). Neural Nets for Free-Text Information Filtering. *Proceedings of the 3rd Australian Conference on Neural Nets*. February 3-5, Canberra, Australia.

[Scholtes, 1992c]: Scholtes, J.C. (1992). Filtering the Pravda with a Self-Organizing Neural Net. *Proceedings of the Symposium on Document Analysis and Information Retrieval*, March 16-18, 1992, Las Vegas, NV, pp. 151-161.

[Scholtes, 1992d]: Scholtes, J.C. (1992). Filtering the Pravda with a Self-Organizing Neural Net. *Proceedings of the First SHOE Workshop*. February 27-28, Tilburg, The Netherlands, pp. 267-277.

[Scholtes, 1992e]: Scholtes, J. (1992). Neural Data Oriented Parsing. *Proceedings of the 3rd Twente Workshop on Language Technology.* May 12-13, Twente, The Netherlands,

[Scholtes, 1992f]: Scholtes, J.C. (1992). Neural Data Oriented Parsing. *Proceedings of the 2nd SNN.* April 14-15, Nijmegen, The Netherlands, pp. 86.

[Scholtes, 1992g]: Scholtes, J.C. (1992). Resolving Linguistic Ambiguities with a Neural Data-Oriented Parsing (DOP) System. *Proceedings of the First SHOE Workshop*, February 27-28, Tilburg, The Netherlands, pp. 279-282.

[Scholtes, 1992h]: Scholtes, J.C. (1992). Resolving Linguistic Ambiguities with a Neural Data-Oriented Parsing (DOP) System. In: *Artificial Neural Networks 2* (I. Aleksander and J. Taylor, Eds.). Vol. 2, pp. 1347-1350. Elsevier Science Publishers.

[Schyns, 1990a]: Schyns, P.G. (1990). A Modular Neural Network Model of the Acquisition of Category Names in Children. In: Connectionist Models: Proceedings of the 1990 Summer School (D.S. Touretzky, J.L. Elman, T.J. Sejnowski, and G.E. Hinton, Eds.), pp. 228-235. Morgan Kaufmann.

[Schyns, 1990b]: Schyns, P.G. (1990). Expertise acquisition through the refinement of a conceptual representation in a self-organizing architecture. *Proceedings of the International Joint Conference on Neural Networks* 1990, Vol. 1, pp. 236-240.

[Schyns, 1991]: Schyns, P.G. (1991). A Modular Neural Network Model of Concept Acquisition. *Cognitive Science*, Vol. 15, pp. 461-508.

[Sejnowski et al., 1986]: Sejnowski, T.J. and Rosenberg, C.R. (1986). NETtalk: A Parallel Network That Learns to Read Aloud. *Technical Report* JHU/EECS-86/01, John Hopkins University.

[Servan-Schreiber et al., 1988]: Servan-Schreiber, D., Cleeremans, A. and McClelland, J.L. (1988). Encoding Sequential Structure in Simple Recurrent Networks. *Technical Report* TR CMU-CS-88-183, Carnegie-Mellon University, PA.

[Servan-Schreiber et al., 1989]: Servan-Schreiber, D., Cleeremans, A. and McClelland, J.L. (1989). Learning Sequential Structure in Simple Recurrent Networks. In: *Advances in Neural Information Processing Systems* (D.S. Touretsky, Editor), Vol. 1, pp. 643-652. Morgan Kaufmann.

[Servan-Schreiber et al., 1991]: Servan-Schreiber, D., Cleeremans, A. and McClelland, J.L. (1991). Graded State Machines: The Representation of Temporal Contingencies in Simple Recurrent Networks. *Machine Learning. Special Issue on Connectionist Approaches to Language Learning.* Vol. 7, No. 2-3, pp. 161-194.

[Shannon et al., 1949]: Shannon, C.E. and Weaver, W. (1949). *The Mathematical Theory of Communication.* Univ. of Illinois Press.

[Shannon, 1948]: Shannon, C.E. (1948). A Mathematical Theory of Communication. *Bell Systems Technical Journal*, Vol. 27, pp. 623-656.

[Shannon, 1951]: Shannon, C.E. (1951). Prediction and Entropy of Printed English. *Bell Systems Technical Journal*, Vol. 30, pp. 50-64.

[Sharkey et al., 1989]: Sharkey, A.J.C., Sharkey, N.E. (1989). Lexical Processing and the mechanism of Context Effects in text Comprehension. *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, August. 16-19, Ann Arbor, MI. pp. 466-473.

[Shinghal et al., 1979a]: Shinghal, R., Toussaint, G.T. (1979). A Bottom-Up and Top-Down Approach to Using Context in text Recognition. *International Journal of Man-Machine Studies*, 1979, pp. 201-212.

[Shinghal et al., 1979b]: Shinghal, R, Toussaint, G.T. (1979). Experiments in Text Recognition with the modified Viterbi Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 184-193.

[Siedlecki et al., 1988]: Siedlecki, W., Siedlecki, K., and Sklansky, J. (1988). An Overview of Mapping Techniques for Exploratory Pattern Analysis. *Pattern Recognition*, Vol. 21, No. 5, pp. 411-429.

[Sikkel, 1990]: Sikkel, N. (1990). Connectionist Parsing of Context-Free Grammars. *Memoranda Informatica* 90-30, Universiteit van Twente, The Netherlands.

[Silverman, 1988]: Silverman, R.H. (1988). Time-Sequential Self-Organization of Hierarchical Neural Networks. In: *Neural Information Processing Systems* (D.Z. Anderson, Editor), pp. 709-714. American Institute of Physics.

[Simard et al., 1988]: Simard, P.Y., Ottaway, M.B. and Ballard, D.H. (1988). Analysis of Recurrent Backpropagation. In: *Proceedings of the Connectionist Models Summer School* (D.S. Touretzky, G.E. Hinton, and T.J. Sejnowski, Eds.), Carnegie Mellon University, PA, pp. 103-113. Morgan Kaufmann.

[Small et al., 1974]: Small, H. and Griffith, B.C. (1974). The Structure of Scientific Literature I: Identifying and Graphing Specialities. *Social Studies*, Vol. 4, pp. 17-40.

[Small et al., 1982]: Small, S., Cottrell, G. and Shastri, L. (1982). Towards Connectionist Parsing. *Proceedings of the National Conference on Artificial Intelligence*, Pittsburg, PA, pp. 247-250.

[Smith et al., 1990]: Smith, K.R. and Miller, M.I. (1990). Bayesian Inference of Regular Grammar and Markov Source Models. In: *Advances in Neural Information Processing Systems* (D.S. Touretsky, Editor), Vol. 2, pp. 388-395. Morgan Kaufmann.

[Smith, 1973]: Smith, R.N. (1973). *Probabilistic Performance Models of Language*. Mouton.

[Smolensky, 1987]: Smolensky, P. (1987). Connectionist AI, Symbolic AI, and the Brain. *Artificial Intelligence Review*, Vol. 1, no. 2, pp. 95-109.

[Smolensky, 1988]: Smolensky, P. (1988). On the proper treatment of Connectionism. *Bahavioral and Brain Sciences*, Vol. 11, pp. 1-23.

[Smolensky, 1990]: Smolensky, P. (1990). Tensor Product Variable Binding and the representation of symbolic Structures in Connectionist Systems. *Artificial Intelligence*, Vol. 46, pp. 159-216. Elsevier Science Publishers.

[Sontag, 1990]: Sontag, E.D. (1990). Feedback Stabilization Using Two-Hidden Layer Nets. *Technical Report* SYCON-90-11, Department of Mathematics, Rutgers University, New Brunswick, NJ.

[Sparck Jones, 1971]: Sparck Jones, K. (1971). *Automatic Keyword Classification for Information Retrieval*, Butterworths.

[Srihari et al., 1983]: Srihari, S.N. and Hull, J.J. (1983). Integrating Diverse Knowledge Sources in Text Recognition. *ACM Transactions on Office Information Systems*, Jan. 1983, pp. 68-87, pp. 216-235.

[Srihari, 1985]: Srihari, S.N. (1985). *Computer Text Recognition and Error Correction: a Tutorial*. IEEE Computer Society.

[St. John et al., 1987]: St. John, M.F. and McClelland, J.L. (1987). Reconstructive Memory for Sentences: A Parallel Distributed Processing Approach. *Proceedings of the Ohio University Infer. Conference*, OH.

[St. John et al., 1988a]: St. John, M.F. and McClelland, J.L. (1988). Applying Contextual Constraints in Sentence Comprehension. In: *Proceedings of the Connectionist Models Summer School* (D.S. Touretzky, G.E. Hinton, and T.J. Sejnowski, Eds.), Carnegie Mellon University, PA, pp. 338-346. Morgan Kaufmann.

[St. John et al., 1988b]: St. John, M.F. and McClelland, J.L. (1988). Applying Contextual Constraints in Sequence Comprehension. *Proceedings of the 10^{th} Annual Conference of the Cognitive Science Society*, Montreal, Canada, pp. 26-35.

[Stanfill et al., 1986]: Stanfill, C.W. and Kahle, B. (1986). Parallel Free-Text Search on the Connection Machine System. *Communications of the ACM*, Vol. 29, pp. 1229-1239.

[Stanfill et al., 1989]: Stanfill, C.W., Thau, R. and Waltz, D.L. (1989). A Parallel Indexed Algorithm for Information Retrieval. *Proceedings of the 12th ACM-SIGIR International Conference on Research & Development in Information Retrieval*, June 26-28.

[Stolcke, 1990]: Stolcke, A. (1990). Learning Feature-Based Semantics with Simple Recurrent Networks. *Technical Report* TR-90-015, april 1990, ICSI Berkeley, CA.

[Stotzka et al., 1990]: Stotzka, R. and Manner, R. (1990). Self-Organization in a Linear Multi-Layered Feedforward Neural Network. *Proceedings of the International Joint Conference on Neural Networks*, January 15-19, Washington DC. Vol. 1, pp. 126-129.

[Sun et al., 1990a]: Sun, G.Z., Chen, H,H,, Lee, Y.C. and Giles, C.L. (1990). Recurrent Neural Nets, Hidden Markov Models and Stochastic Grammars. *Proceedings of the IEEE International Joint Conference on Neural Networks*, June 17-21, San Diego, CA. Vol. 1, pp. 729-734.

[Sun et al., 1990b]: Sun, G.Z., Chen, H.H., Giles, C.L., Lee, Y.C. and Chen, D. (1990). Connectionist Pushdown Automata that Learn Context-Free Grammars. *Proceedings of the International Joint Conference on Neural Networks*, January 15-19, Washington DC, Vol. 1, pp. 577-580.

[Sutton et al., 1981]: Sutton, R.S. and Barto, A.G. (1981). Towards a Modern Theory of Adaptive Networks: Expectation and Prediction. *Psychological Review*, Vol. 88, pp. 135-170.

[Tanaka, 1990]: Tanaka, S. (1990). Theory of Self-Organization of Cortical Maps: Mathematical Framework. *Neural Networks*, Vol. 3, No. 6, pp. 625-640.

[Tavan et al., 1990]: Tavan, P., Grubmller, H. and Khnel, H. (1990). Self-Organization od Associative Memory and Pattern Classification: Recurrent Signal Processing on Topological Feature Maps. *Biological Cybernetics*, Vol. 64, pp. 95-105.

[Tenopir et al., 1990]: Tenopir, C. and Ro, J.S. (1990). *Full Text Databases*. Greenwood Publishing Group.

[Tenopir, 1984]: Tenopir, C. (1984). Full-Text Databases. In: *Annual Review of Information Science and Technology* (Ed. William, N.), Vol. 19, pp. 215-246.

[Thacker et al., 1990]: Thacker, N.A. and Mayhew, J.E.W. (1990). Designing a Layered Network for Context Sensitive Pattern Classification. *Neural Networks*, Vol. 3, No. 3, pp. 291-299.

[Touretzky et al., 1985]: Touretzky, D.S. and Hinton, G.E. (1985). Symbols among Neurons. *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pp. 238-243.

[Touretzky et al., 1988]: Touretzky, D.S. and Hinton, G.E. (1988). A Distributed Connectionist Production System. *Cognitive Science*, Vol. 12, Number 3, September, pp. 423-466.

[Touretzky, 1986]: Touretzky, D.S. (1986). BolzCONS. *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, pp. 522-530.

[Touretzky, 1987]: Touretzky, D.S. (1987). Representing Conceptual Structures in a Neural Network. *Proceedings of the IEEE International Neural Network Conference*, June 21-24, San Diego, Vol. 2, pp. 279-286.

[Touretzky, 1990]: Touretzky, D.S. (1990). BoltzCONS: Dynamic Symbol Structures in a Connectionist Network. *Artificial Intelligence*, Vol. 46, pp. 5-46. Elsevier Science Publishers.

[Tsung et al., 1989]: Tsung, F.-S. and Cottrell, G.W. (1989). A Sequential Adder Using Recurrent Networks. *Proceedings of the International Joint Conference on Neural Networks*, Vol. 2, pp. 133-139.

[Tsung, 1990]: Tsung, F.-S. (1990). Learning in Recurrent Finite Difference Networks. In: *Connectionist Models: Proceedings of the 1990 Summer School* (D.S. Touretzky, J.L. Elman, T.J. Sejnowski, and G.E. Hinton, Eds.), pp. 124-130. Morgan Kaufmann.

[Van Opdorp et al., 1991]: Van Opdorp, G.J. (1991). Networks at Work: A Connectionist Approach to Non-Deductive Legal Reasoning. *Proceedingsof the International Conference on Artificial Intelligence in Law.*

[Von Neumann, 1945/1982]: Von Neumann, J. (1945). First Draft of a Report on the EDVAC. In: *The Origins of Digital Computers: Selected Papers* (B. Randall, Editor), Berlin: Springer.

[Voorhees, 1985]: Voorhees, E.M. (1985). The Cluster Hypothesis Revisited. *Proceedings of the 8th Annual International ACM-SIGIR Conference on Research & Development in Information Retrieval*, June '85, pp. 188-196.

[Wagner et al., 1974]: Wagner, R.A. and Fischer, M.J. (1974). The String-to-String Correction Problem. *Journal of the ACM*, Vol. 21, No. 1, pp. 168-173.

[Waltz et al., 1984]: Waltz, D.L. and Pollack, J.B. (1984). Parallel Interpretation of Natural Language. *Proceedings of the International Conference on Fifth Generation Computer Systems* 1984, ICOT.

[Waltz et al., 1985]: Waltz, D.L. and Pollack, J.B. (1985). Massively Parallel Parsing. *Cognitive Science*, Vol. 9, Number 1, pp. 51-74.

[Weber et al., 1990]: Weber, S.H. and Stolcke, A. (1990). L0: A Testbed for Miniature Language Acquisition. *Technical Report* TR-90-010, July 1990, ICSI Berkeley, CA.

[Weijters, 1990]: Weijters, A.J.M.M. (1990). NetSpraak: A Grapheme-to-Phoneme Conversion Network for Dutch. *Proceedings of the IEEE Symposium on Neural Networks*, June 21, Delft, Netherlands, pp. 59-68.

[Werbos, 1974]: Werbos, P. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. *Doctoral Dissertation Applied Mathematics*, Harvard University.

[Wermter, 1991]: Wermter, S. (1991). Learning to Classify Neural Language Titles in a Recurrent Connectionist Model. In: *Artificial Neural Networks* (T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, Eds.), Vol. 2, pp. 1715-1718. Elsevier Science Publishers.

[Wernicke, 1908]: Wernicke, (1908). The Symptom-Complex of Aphasia. In: *Diseases of the Nervous System* (A. Church, Editor), pp. 265-324. Appleton.

[Wettler et al., 1989]: Wettler, M., Rapp, R. (1989). A Connectionist System to Simulate Lexical Decisions in Information Retrieval. In: *Connectionism in Perspective* (R. Pfeiffer et al., Eds.), pp. 463-469. North-Holland.

[Wettler et al., 1990]: Wettler, M., Rapp, R. (1990). Parallel Associative Processes in Information Retrieval. In: *Parallel Processing in Neural Systems and Computers* (R. Eckmiller, G. Hartmann and G. Hauske, Eds), pp. 509-512. Elsevier Science Publisher

[Widrow et al., 1960]: Widrow, B. and Hoff, M.E. (1960). Adaptive Switching Circuits. *1960 WESCON Convention*, Record Part IV, pp. 96-104.

[Wiener, 1948/1961]: Wiener, N. (1961). *Cybernetics (Second Edition)*. MIT Press.

[Wilbert, 1991]: Wilbert, R. (1991). Assoziative Begriffsrepresentation in Neuronalen Netzen. *Nachrichten für Dokumentation*, Vol. 42, No. 3, pp. 205-211.

[Wilkinson et al., 1991]: Wilkinson, R. and Hingston, P. (1991). Using the Cosine Measure in a Neural Network for Document Retrieval. *Proceedings of the 14th ACM-SIGIR Conference on Research & Development in Information Retrieval*, October 13-16. Chicago, MI, pp. 202-210.

[Willett, 1979]: Willett, P. (1984). Document Retrieval Experiments Using Indexing Vocabularies of Varying Size, II. Hashing, Truncation, Digram and Trigram Encoding of Index Terms. *Journal of Documentation*, Vol. 35, No. 4, pp. 215-246.

[Willett, 1984]: Willett, P. (1984). A Note on the Use of Nearest Neighbors for Implementing Single Linkage Document Classification. *Journal of the ASIS*, Vol. 35, No. 3, pp. 149-152.

[Willett, 1988]: Willett, P. (1988). Recent Trends in Hierarchical Document Clustering: A Critical Review. *Information Processing and Management*, Vol. 24, No. 5, pp. 577-597.

[Williams et al., 1988]: Williams, R.J. and Zipser, D. (1988). A Learning Algorithm for Continually Running Fully Recurrent Networks. *Technical Report* ICS Report 8805, UCSD, La Jolla, CA.

[Williams et al., 1989a]: Williams, R.J. and Zipser, D. (1989). Experimental Analysis of the Real-Time Recurrent Learning Algorithm. *Connection Science*, Vol. 1, No. 1, pp. 87-111.

[Williams et al., 1989b]: Williams, R.J. and Zipser, D. (1989). A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, Vol. 1, pp. 270-280.

[Winston, 1975]: Winston, P.H. (1975). Learning Structural Descriptions from Examples. In: *The Psychology of Computer Vision,* (P.H. Winston, Editor), McGraw-Hill Book Company.

[Woods, 1970]: Woods, W.A. (1970). Transition Network Grammars for Natural Language Analysis. *Communications of the ACM*, Vol. 3 nr. 10, pp. 591-606.

[Yen et al., 1990]: Yen, M.M., Blackburn, M.R. and Nguyen, H.G. (1990). Feature Maps based Weight Vectors for Spatiotemporal Pattern Recognition with Neural Nets. *Proceedings of the International Joint Conference on Neural Networks*, June 17-21. San Diego CA, Vol. 2, pp. 149-155.

Nederlandstalige Samenvatting

Dit proefschrift handelt over machine intelligentie (met zelf-organiserende neurale netwerken in het bijzonder), natuurlijke taalverwerking en information retrieval.

De huidige generatie generatieve grammatica's die gebruikt worden in de computationele linguistiek zijn niet in staat om natuurlijke taal, zoals mensen die plegen te gebruiken, adequaat te verwerken. Een aantal van deze problemen hebben betrekking tot het niet kunnen verwerken van fout of verminkt taalgebruik, het niet kunnen corrigeren van verkeerd taalgebruik, het niet kunnen generaliseren en het niet effectief kunnen combineren van syntactische, semantische en statistische eigenschappen van taal.

Op het gebied van de *information retrieval* gelden een aantal andere problemen. Door de grote hoeveelheden tekst is men bij het zoeken in zeer grote tekst databases in veel gevallen niet in staat meer te implementeren dan een globale oppervlakte analyse. Iedere andere vorm van contextuele analyse brengt grote computationele problemen met zich mee. Het moge duidelijk zijn dat voor de correcte interpretatie van de inhoud van een document meer vereist is.

Als tegenhanger van de traditionele generatieve grammatica's in de computationele linguistiek worden een aantal modellen gepresenteerd die wel degelijk in staat zijn tot het efficiënt en robuust verwerken van natuurlijke taal. In plaats van taal te verwerken aan de hand van regels worden zinnen geanalyseerd door ze (recursief) te vergelijken met bestaande (deel) zinnen in een groot geanalyseerd corpus: het zogenaamde *data oriented parsing* (DOP) paradigma. Hierdoor is het systeem beter in staat om te gaan met verkeerde zinnen en idiomatische uitzonderingen. Het systeem vertoont zelfs enig creatief gedrag. De rol van de neural netwerken in het systeem ligt in de effectieve context-afhankelijke implementatie van het corpus.

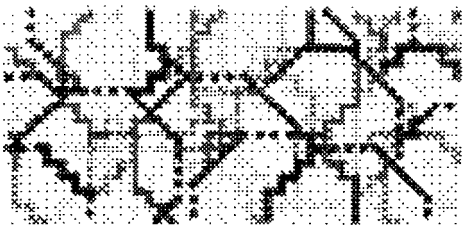
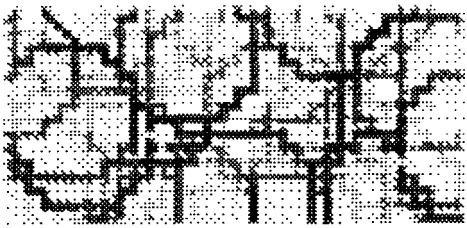
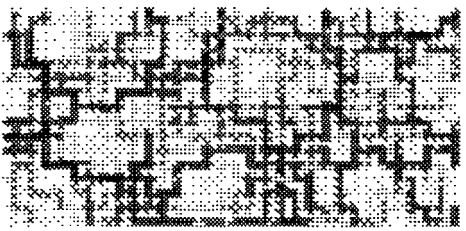
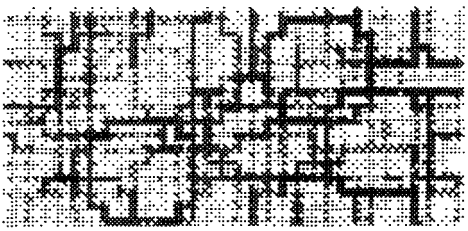
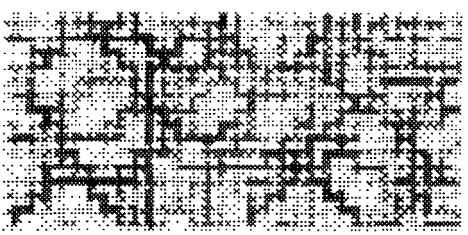
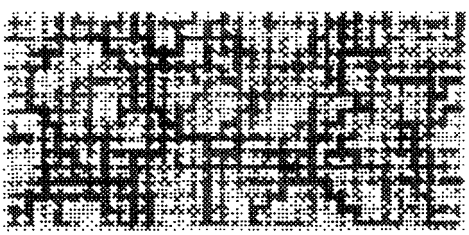
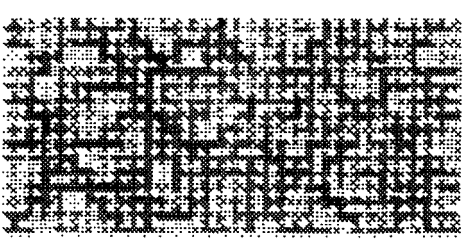
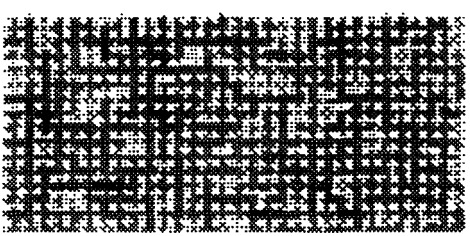
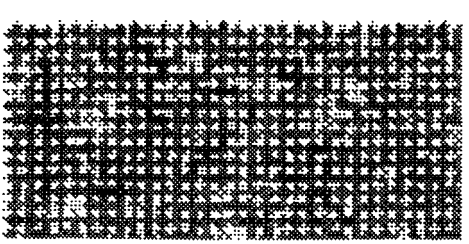
Als variant op de oppervlakkige index-technieken in information retrieval maken de hier gepresenteerde technieken het mogelijk om meer context te implementeren zonder aan snelheid in te boeten. De rol van de neurale netwerken ligt hier vooral in de automatische afleiding van context-afhankelijke waarschijnlijkheden in een real-time filter omgeving.

Voortbouwend op corpus-gebaseerde technieken en zelf-organiserende neurale netwerken worden in dit werk een aantal mogelijke oplossingen voor problemen uit de traditionele computationele linguistiek en information retrieval gepresenteerd. Toekomstig onderzoek kan bestaan uit de toepassing van dit soort technieken in case-based reasoning en real-time knowledge based systems zoals spraakherkenning en optical character recognition (OCR).

Curriculum Vitae

Naam:	Johannes Cornelis Scholtes
Adres:	Dufaystraat 1, 1075 GR Amsterdam
Geboren:	7 juni 1964 te Leiden

1970-1976	Basisschool (R.K. Banneux School te Hazerswoude)
1976-1982	Atheneum B (R.K. Adelbert College te Wassenaar)
1982-1987	Informatica ingenieur (Technische Universiteit te Delft)
1987	Oprichting M.S.C.
1988-1989	Koninklijke Marine
sedert 1989	Verbonden aan de vakgroep alfa-informatica van de Universiteit van Amsterdam.

M. S. C.
Information
Retrieval
Technologies