LEARNED TEXT CATEGORIZATION BY BACKPROPAGATION NEURAL NETWORK

BY

LAM LAI YIN, DOMINIC SAVIO

A Thesis Presented to
The Hong Kong University of Science and Technology
In Partial Fulfillment
of the Requirements for
the Degree of Master of Philosophy
in Computer Science

Hong Kong, August 1996

Authorisation

I hereby declare that I am the sole author of the thesis.

I authorise The Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorise The Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

LEARNED TEXT CATEGORIZATION BY BACKPROPAGATION NEURAL NETWORK

BY

LAM LAI YIN, DOMINIC SAVIO

APPROVED:

____

DR. DIK LUN LEE, SUPERVISOR

____

PROF. ROLAND T. CHIN, HEAD OF DEPARTMENT

Department of Computer Science

27 August 1996

DEDICATION

To my parents who have given me the greatest love and care,
and to Mandy who has given me the warmest support.

ACKNOWLEDGEMENTS

I would like to express my greatest thanks to my academic advisor Dr. Dik-Lun Lee for his valuable advices and willingness to let me explore, without which this thesis would not have been completed. I thank every member of the information retrieval research group in the Computer Science Department at the Hong Kong University of Science and Technology. Through our discussions, I have gained much both as a researcher and in my personal development. I thank Dr. Dit-Yan Yeung who enlightened me on the ideas of principal component analysis and made me aware of the possibility of applying it for dimensionality reduction. I also thank Dr. Yeung and Dr. Beat Wüthrich for being my thesis committee members.

This research was supported by the Research Grant Council of Hong Kong under the project number HKUST670/95E. I thank Reuters Ltd. and Carnegie Group Inc. for making the Reuters-22173 test collection freely available for research purposes, and all who have involved in reformatting and enhancing the test collection.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LEARNED TEXT CATEGORIZATION BY BACKPROPAGATION NEURAL NETWORK

BY

LAM LAI YIN, DOMINIC SAVIO

A Thesis Presented to
The Hong Kong University of Science and Technology
in Partial Fulfillment
of the Requirements for
the Degree of Master of Philosophy

Hong Kong, August 1996

ABSTRACT

Text categorization is the classification of unstructured text documents with respect to a set of one or more pre-defined categories. This task is often performed in automatic text indexing systems to assign subject categories to text documents. The benefit of text categorization is that once the documents are categorized, users can limit the scope of search by concentrating on a few categories relevant to their information needs.

In this thesis, we propose a text categorization model using an artificial neural network trained by the Backpropagation learning algorithm as the text classifier. Due to the high dimensionality of the feature space typical for textual data, scalability is poor if the neural network is trained using this high dimensional raw data. In order to improve the scalability of the proposed model, we proposed and compared four dimensionality reduction techniques to reduce the feature space into an input space of much lower dimension for the neural network classifier. The first three of these techniques are domain dependent term selection methods, namely the DF method, the CF-DF method and the TFxIDF method.

The fourth technique is a domain independent feature extraction method based on a statistical multivariate data analysis technique called Principal Component Analysis.

To test the effectiveness of the proposed model, experiments were conducted using a subset of the Reuters-22173 test collection for text categorization. The results showed that the proposed model was able to achieve high categorization effectiveness as measured by precision and recall. Among the four dimensionality reduction techniques proposed, Principal Component Analysis was found to be the most effective in reducing the dimensionality of the feature space.

CHAPTER 1

INTRODUCTION

In the last few years, there has been enormous growth in the amount of text documents available in digital form. Part of the driving force behind this is the advance in document imaging technologies, which facilitate the conversion of paper documents into digital form, and the popularity of word processing software enabling the creation of digital documents. The popularity of the Internet recently has only accelerated this growth. Internet services such as electronic mail, USENET News, and the World Wide Web are still mainly text based. Their popular usage has resulted in millions of text documents being generated on a daily basis. This has two implications on the user:

1. The availability of information becomes less of a problem as more and more information becomes available online. In particular, the World Wide Web has enabled access to a large amount of distributed information for anyone who has Internet access.

2. What is becoming more of a problem with the growth in the amount of digitally stored textual information is a phenomenon known as information overloading. In other words, there is so much information available that it becomes increasingly difficult and time-consuming for the user to find the information relevant to his needs.

While any reasonable database management system allows straightforward access to various types of structured data, locating the relevant information according to the information need of the user can be very difficult for unstructured information. In view of this, there has been much research done by the information retrieval (IR) community investigating ways to enable faster and easier access to relevant unstructured information, especially unstructured text.

The explosive growth of unstructured information on the Internet and in particular the World Wide Web has greatly increased the need for information retrieval systems. This is evident from the rapid growth of the number of World Wide Web indexing services which have become available recently. Some of the most popular World Wide Web indexing services include Alta Vista, Infoseek, Lycos, Inktomi, and Yahoo.

In this chapter, we will give a brief overview of a number of text-processing tasks that has been the focus of information retrieval researchers. We will explain our motivations in choosing one of the text-processing tasks, namely text categorization as the focus of our research, and the objectives we would like to achieve. An overview of the outline of this thesis is given at the end of this chapter.

1.1 Text-processing Tasks

A number of major text-processing tasks has been of interest to information retrieval researchers. These include text retrieval, text routing or filtering and text categorization. In this section, we will give a brief description of these tasks. For each of the text-processing task, the basic information unit to be processed is a text document. Thus, the terms "information" and "document" will be used interchangeably throughout the remaining parts of this thesis.

1.1.1 Text Retrieval

Given a text document database containing a large number of text documents, a text retrieval engine has to select a subset of the documents to be returned or displayed to the user. The selection is usually based on an input query, representing the information needs of the user. In keyword based text retrieval, the query can be a set of keywords selected by the user as representative to his information needs. The selection criterion can be binary, by which a document is either selected or not selected, or it may involve sorting the documents according to some scoring scheme based on the estimated relevance of each document to the query, and selecting only a subset of the documents with relatively higher scores than others. The process of sorting the selected documents based on their relevance scores is commonly known as document ranking.

In some text retrieval systems, a mechanism called *relevance feedback* [12, 52] is provided. In this mechanism, the user provides feedback to the system by indicating which of the retrieved documents he think are relevant to his needs. After the user has marked a set of documents as relevant, these documents are analyzed to select a set of keywords. These selected keywords are added to the original query to form an augmented query. This augmented query is then used for another retrieval with the aim that more relevant documents will be retrieved. This process can be repeated for a number of times until the user is satisfied with the retrieval result.

1.1.2 Text Filtering

Text filtering can be described as the selective dissemination of textual information. Like text retrieval, text filtering involves the selection of a subset from the available text documents, according to the information needs of the user. In text filtering, the information need of the user is typically represented as a user profile, which can be a set of keyword-based queries specified by the user. However, unlike text retrieval, where the information needs of the user are mostly short term, the user profile used for text filtering represents the long term information needs of the user.

In text retrieval, the retrieval process begins by the user typing in a query for his information needs, and ends when the result is returned to the user. If the user has other information needs, he must begin another retrieval process by typing in another query. The document database from which documents are retrieved is created before any retrieval session begins, and serves as an archive of available text documents on which multiple retrievals are carried out.

On the other hand, a typical text filtering system processes an incoming stream of new documents, which is often done in real time. The filtering system decides whether or not a document is to be disseminated to the user according to the user profile. Once the documents are disseminated to the user, they are either deleted from the system or passed to the document database of a text retrieval system. While the user profile can be edited by the user, it usually represents long term information needs and may remain unchanged for a long period of time. In the case when a filtering system is serving multiple users, the filtering decision is made based on the user profile corresponding to each user.

1.1.3 Text Categorization

Text categorization is the classification of text documents into a set of one or more categories. In text categorization, the categories that the documents are to be classified into are pre-defined, typically by the designer or maintainer of the categorization system. The end users are usually not involved in the process of defining the categories. Nevertheless, once the documents are classified into categories, users can identify a number of categories that may contain documents relevant to his needs, and ignore documents in categories that are unlikely to be relevant. In this way, the scope of information that the user has to search is largely reduced, which in turn speeds up the process of finding relevant information for the user. In fact, a typical application of text categorization systems is to limit the scope of searching for text retrieval systems. In addition to specifying a query, the user can limit the scope of search by specifying a set of categories to be searched.

Depending on the set of categories, there may be overlaps between different categories, in which a document can belong to more than one categories. In some cases, some documents may belong to none of the defined categories. Depending on the categorization system, these documents may be removed or put into a special "unclassified" category. Like user profiles in text filtering systems, the set of categories is usually unchanged over a long period of time.

Similar to text filtering, text categorization systems are typically required to process a stream of incoming new documents, sometimes in real time. During the categorization process, documents are assigned zero, one, or more categories. The processed documents are then stored in a document database together with the list of category labels they are assigned to. As in text retrieval, the membership of a document to a category can be binary, in which a document is either assigned or not assigned a category. In the case of graded categorization, each document is given scores indicating its relevance to each of the categories. If binary membership is to be obtained from a graded categorization system, a threshold can be set so that a category is assigned to a document only if the relevance score corresponding to the category is above the threshold.

1.2 Motivation and Objectives

Among the text-processing tasks, text retrieval has attracted the most research effort in the IR community. As a result, the problem is well studied and some well recognized and proven text retrieval models have emerged, such as the vector space model proposed by Salton [50]. On the other hand, text filtering and text categorization are much less studied as compared to text retrieval. While there are a number of models proposed by various researchers for these two tasks, no single model has yet gained general recognition by the IR community. It is thus our belief that more research efforts should be placed on the study of the latter two tasks.

The objective of this thesis is to investigate the effectiveness of neural network learning techniques as applied to the information retrieval domain. In order to take advantage of the neural network model of computation, our methodology is to formulate the text-processing tasks as a learned classification problem. We have chosen to place our focus on text categorization as we observe that among the three text-processing tasks text categorization fits the learned classification problem the best:

1. As already mentioned in the last section, in text categorization, the set of categories is usually pre-defined before the development of the categorization system, and remains unchanged for a long period of time. This

is essential as the number and types of the categories can be taken into account in choosing the topology and size of the neural network classifier.

2. When a categorization system is developed, often there will be a considerable number of documents that have already been categorized manually. These manually categorized documents can be used as training examples for training the neural network classifier.

3. Unlike text categorization where the categories are pre-defined, the set of queries input by the users is unknown for a text retrieval system. As a result, a text retrieval system is assumed to be able to handle any possible queries input by the users. Due to the huge number of possible queries, it is impractical to consider each of the queries as a single class and try to construct a classifier for all of these classes. While we can formulate the text retrieval task as a classification problem consisting of two classes: documents that are relevant to the information needs of the user and those that are not, and feed the user query together with the documents as input to the classifier, finding a set of training examples extensive enough to ensure good performance on any possible query is practically impossible.

4. For text filtering, we can formulate the task as a classification problem by considering each user profile as a different class. However, the number of user profiles is continuously changing as new users are added and existing users quit using the system. As a result, the number of classes is unknown when the classifier is developed. One possible solution is to construct a separate neural network for each of the individual users. However, it still depends on the willingness of the user to provide enough training examples during the profile creation process to adequately train the neural network. Moreover, the ability of a general user to provide adequate training examples for high quality training is questionable.

As in many other classification problems, neural network classifiers trained with raw data do not scale well due to the high dimensionality of the raw data. This is especially true for information retrieval tasks as each unique indexing term represents a new dimension. One common solution to the scaling problem is to encode the high dimensional raw input data in a much lower dimensional feature space. This encoding process is often known as input dimensionality reduction.

In order to improve the scalability of the text categorizer, we have investigated a number of feature selection and extraction techniques as a preprocessing step for reducing the input dimensionality of the neural network classifier. Among these techniques, emphasis has been placed on a statistical technique called Principal Component Analysis (PCA) [22, 23].

Due to the relative success and popularity of Backpropagation neural network training in various domains, we have chosen to focus on this technique for training our neural network classifier.

1.3 Outline of the Thesis

The rest of this thesis is divided into a number of chapters. In chapter 2, an introduction to artificial neural networks will be given. For readers who have no previous background in artificial neural networks, this chapter serves as a foundation to the topic. In chapter 3, we will give a summary of some of the work done on the application of neural networks to information retrieval. Chapter 4 is an introduction to a statistical data analysis technique called principal component analysis. Focus will be placed on the application of principal component analysis for dimensionality reduction. In chapter 5, we will discuss the details of the text categorization model we have proposed. In chapter 6, procedures for performance evaluation of information retrieval systems will be discussed. We will also present and discuss the results of the experiments we have conducted to evaluate the performance of the proposed categorization model. Chapter 7 is the concluding chapter. We will summarize the contributions of this research and pinpoint some of the ways that the research can be extended in the future within this chapter.

CHAPTER 2

ARTIFICIAL NEURAL NETWORKS AND BACKPROPAGATION LEARNING

In this chapter, an introduction to artificial neural networks and an overview to the Backpropagation learning rule is given. For readers who have no background knowledge on Backpropagation training in feed-forward neural networks, this chapter serves as a foundation for understanding the materials discussed in later chapters. Readers who have already had adequate background in the topic may choose to skim through the chapter quickly.

2.1 Origin of Artificial Neural Networks

Artificial neural networks, also known as connectionist models or parallel distributed processing, can be seen as a class of computational models or algorithms which works by imitating the working mechanism of the human brain, despite the fact that there may be oversimplification in many cases. Earlier work in neural network research dated back to the 1940s, when McCulloch and Pitts described a simplified model of a neuron [36]. Over the years, there have been a lot of proposals on different neural computation models. However, there are some basic elements common to most of these models. In the following sections, we will try to summarize these elements with the aim of giving the readers a better picture of what neural networks are from the computer science perspective.

2.2 Definition of an Artificial Neural Network

An artificial neural network consists of a large number of simple processing units, which are often referred to as neurons. Neurons are connected to form a network by weighted connections, or synapses, through which neurons can communicate by sending signals to each other. By the local computations performed within each neuron and the signals sent between them through the connections, the neural network as a whole performs computation in a highly parallel and distributed manner.

For our discussion, we will assume all connections are directed. That is, signal must always flow in only one direction through a connection, from the output of the presynaptic neuron to the input of the postsynaptic neuron. If signals are flowing between two neurons in both directions, we assume there are two parallel but oppositely directed connections between the two neurons.

2.3 Neural Network Topologies

In many networks, the neurons are arranged in a number of layers. Neural networks having more than one layer of neurons are sometimes called multi-layer neural networks. Figure 1 shows an artificial neural network with three layers of neurons.

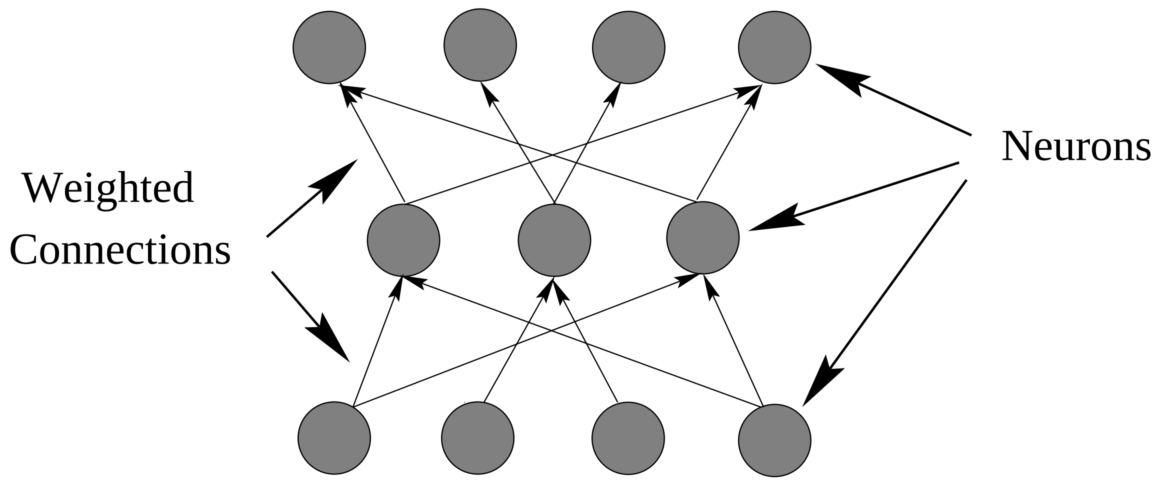

Figure 1: An artificial neural network.

Based on the way the neuron layers are connected together in a neural network, or the network topology, we can broadly classify the various neural networks into two classes: feed-forward networks and recurrent networks.

2.3.1 Feed-forward Networks

In a multi-layer feed-forward neural network, the direction that the signals flow through the connections is strictly feed-forward, from one layer to a following layer. Although the signal flow can continue across multiple layers of neurons, there must not exist any feedback connections, going from one layer to a preceding layer. Moreover, there should not be any intra-layer connections between the neurons of the same layer. Since loops are not possible in a feed-forward neural network, the number of times that the signals will flow in a network is finite, limited by the number of layers or depth of the network. After these finite number of steps of signal flow, all the activation values in the network will be known.

In a feed-forward network, there is usually a layer called the input layer, which accepts external signals as input to the neural network. Also, there is one other layer called the output layer in which the activation values of the neurons in this layer will be taken as output of the neural network. The layers in between the input layer and the output layer are sometimes called hidden layers. Consequently, neurons in the input layer are called input units, while those in the output layer and a hidden layer are called output units and hidden units, respectively.

Some examples of feed-forward neural networks are Perceptrons [46], Adaline [61], and non-linear feed-forward networks trained by Backpropagation [47], which will be discussed in more detail later in this chapter.

2.3.2 Recurrent Networks

As the name suggests, recurrent neural networks are networks that contain feedback connections. There are many types of recurrent neural networks proposed in the literature, including the well-known model proposed by Hopfield [20], often called the Hopfield Network.

Unlike feed-forward networks, there are loops in recurrent networks. As a result, signal flow may continue to any number of steps. In this case, the network is said to undergo a relaxation process until the activation values converge to a stable state. In some cases, this stable state may not be reached and the network will continue to change its activation values without becoming stable. When the network does become stable, the set of activation values at the stable state represents the output of the network.

2.4 The Neuron as a Simple Processing Unit

As described previously, a neuron is a processing unit capable of performing simple computations locally. It receives input for computation from the incoming connections. The result of the computation is sent as output to other neurons through the outgoing connections.

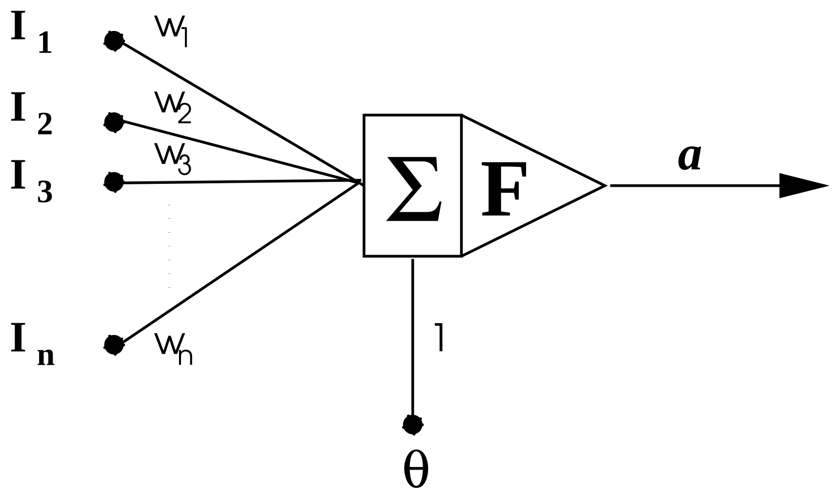

Figure 2: A neuron with multiple input connections.

Figure 2 shows a neuron with n weighted input connections. The computation performed within a neuron involves first finding the net input (net) to the neuron. In most neural networks, this net input is simply the weighted sum of all input signals to the neuron. Usually, there will be an additional input (θ) with weight equals to 1, called the bias. The weighted sum net in mathematical form is:

$$net = \sum_{i=1}^{n} w_i I_i + \theta$$

We define a function F called the activation function of the neuron. The output or activation value (a) of the neuron is computed by passing the net input (net) as an argument to the activation function F:

$$a = F(net)$$

While this computation is standard in most neural networks, different networks can differ in the activation function (F) used and the presence or absence of the bias input (θ). In most cases, the activation function is a nondecreasing function. When the activation function is linear, the neuron is known as a linear unit. On the other hand, the neuron is known as a non-linear unit if a non-linear function is used as the activation function.

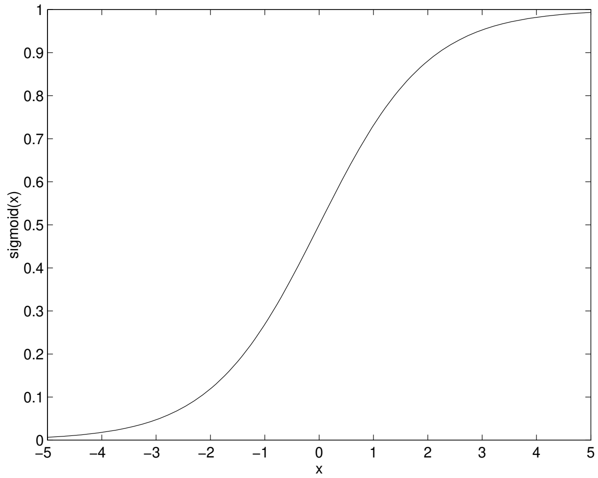

Figure 3: The sigmoid function.

The most commonly used non-linear activation function is the sigmoid function:

$$a = F(net) = \frac{1}{1 + e^{-net}}$$

The sigmoid function is a non-linear, nondecreasing and differentiable function, with output values in the range [0, 1]. As we shall see later, the sigmoid function being differentiable is a requirement for it to be used in a neural network trained with the Backpropagation learning rule. Figure 3 shows a plot of the sigmoid function.

2.5 Learning in Artificial Neural Networks

Up to now, we have not mentioned how learning is done in a neural network. In fact, the ability to learn or adapt is often considered one of the most important properties that has attracted so much interest in neural networks. In this section, we will look at the learning process in neural networks. In order to pave the way for the later discussion of Backpropagation learning in feed-forward neural networks, the focus in this section will be on supervised learning. The training of Adaline using the least mean square (LMS) learning rule or the delta rule will be given as an example.

2.5.1 The Meaning of Learning in Artificial Neural Networks

In a computer program, or any computational modules in general, including artificial neural networks, the ability to learn can be defined as the ability of the module to adjust itself such that the self-adjustment will enable the module to perform a given task or a set of tasks with improved performance over time.

While the above definition is by no means the most complete nor the most accurate one, it does indicate what is happening when an artificial neural network is said to be going through the learning process. That is, when an artificial neural network is trying to learn to perform better on some tasks, it does so by making some changes internally. For some networks, these changes may involve changing the structure or topology of the network, such as adding or removing neurons and connections. But for most of the neural networks existing today, including the neural networks we are concerned with in the rest of this thesis, changes induced by the learning process are limited to the adjustment of the weights corresponding to the network connections.

In order to achieve the effect of learning, the neural network must make changes to its connection weights according to some learning rule. The act of adjusting the connection weights of a neural network by following some learning rule so that the network can learn is called training. There are different learning rules for different types of neural networks. Examples of learning rules are the delta rule for training Adaline [61], and the Backpropagation learning rule for training multi-layer feed-forward neural networks [47].

2.5.2 Different Learning Paradigms

Depending on whether an external teacher is present, we can have two different learning paradigms:

Supervised Learning

Supervised learning refers to the learning paradigm in which an external teacher is present. The role of this external teacher is to give feedback to the learning module about the appropriateness of its performance with respect to some given tasks or goals. In the case of neural network learning, this means that the network is given a set of input-output pairs as training examples. In order to learn, the neural network tries to make adjustment of its connection weights according to some learning rule, such that when given any of the training inputs, the network will eventually give outputs that match the corresponding training outputs.

Unsupervised Learning

In unsupervised learning, there is no external teacher giving feedback to the network. In this paradigm, the neural network makes adjustments to itself based only on a set of input patterns given to it. In other words, a training example given to the network consists of only an input. One common application which often makes use of unsupervised learning in neural networks is in clustering a set of input patterns [26]. In this case, the network learns to discover correlations and regularities in the inputs, and adjusts its response with respect to the inputs accordingly. Unsupervised learning is sometimes referred to as self-organization.

In this thesis, we are only interested in the supervised learning paradigm in neural networks, as it is the learning paradigm for the Backpropagation learning rule on which our text categorization model is based.

2.5.3 Training Adaline with the Delta Rule

In this section, we will introduce the delta rule, which is the learning rule for training a neural network proposed by Widrow and Hoff called the Adaline [61]. It is important to introduce the delta rule here, since the Backpropagation learning rule, which will be discussed in the next section, is actually a generalization of the delta rule for training non-linear multi-layer feed-forward neural networks.

The Adaline

An Adaline, which stands for ADAptive LINear Element, is a single layer neural network with neurons that are linear units. The neurons in an Adaline fit well into the general neural model shown in figure 2, with the activation function (F) being the linear function $F(x) = x$. In other words, the activation value (a) can be expressed as:

$$a = \sum_{i=1}^{n} w_i I_i + \theta$$

which is just the net input to the neuron.

The Learning Rule

The learning paradigm for the Adaline is supervised learning, which means that a set of input-output pattern pairs will be given to the Adaline as training examples. The objective of training is such that given any input pattern, the Adaline will eventually learn to give output that matches the corresponding output pattern. This learning is achieved by making adjustments to the connection weights of the Adaline according to some learning rule.

A first step of deriving a learning rule for training the Adaline, we must have some measures of how close the output given by the Adaline is to the desired output pattern. For this, Widrow and Hoff proposed the least mean square (LMS) error as a measure of the distance between the Adaline's output and the desired output pattern. Thus, the total error (E) summed over all training patterns is:

$$E = \sum_{p=1}^{M} E_p = \frac{1}{2}\sum_{p=1}^{M}(d_p - a_p)^2$$

where p indices over all training patterns, M is the total number of training patterns in the training set, d_p represents the p^{th} desired output pattern, and a_p is the Adaline's output when the p^{th} input pattern is present to it.

The learning rule modifies the connection weights of the Adaline in order to minimize the total error (E). One simple method for error minimization is called gradient descent. Imagine a multi-dimensional space where each dimension corresponds to a connection weight of the Adaline, a plot of the error (E) in this space will form an error surface such that its height at any point is equal to the error (E) given the corresponding set of weights. In this case, gradient descent corresponds to performing a steepest descent on this error surface.

Mathematically, a gradient component is defined as the first partial derivative of the error measured on the current pattern (E_p) with respect to each weight (w_i):

$$\begin{aligned}\frac{\partial E_p}{\partial w_i} &= \frac{\partial E_p}{\partial a_p}\frac{\partial a_p}{\partial w_i} \\ &= -(d_p - a_p)I_{pi}\end{aligned}$$

where I_{pi} is the i^{th} input component of the p^{th} input pattern.

In order to minimize the total error (E) by steepest descent on the error surface, the weights (w_i) must be updated on presentation of each pattern in opposite direction to that of the gradient, that is:

$$\Delta_p w_i = -\eta\frac{\partial E_p}{\partial w_i}$$

$$= \eta(d_p - a_p)I_{pi}$$

where η is a pre-defined constant called the learning rate, which controls the step size of weight update. A large learning rate will accelerate the learning process, but a learning rate that is too large may lead to a phenomenon known as oscillation, which will prevent the minimum error point from being reached. If we let $\delta_p = d_p - a_p$, the previous equation can be written as:

$$\Delta_p w_i = \eta \delta_p I_{pi}$$

This equation forms the rule for weight update in Adaline training, and is known as the *delta rule*. In practice, the weights of the Adaline are initialized to some random values at the beginning. Then, the delta rule is applied repeatedly during training to update the weights until the total error (E) falls below some threshold or tolerance level. The total error (E) is often called the learning error.

2.6 Learning by Error Backpropagation

It was pointed out by Minsky and Papert [38] that single layer linear networks are limited in the kind of tasks that they can perform. Although they had stated that these limitations can be overcome by using a multi-layer neural network, a learning rule was not provided for adjusting the weights of the network. This problem was left unsolved until Rumelhart, Hinton and Williams [47] proposed a generalization of the delta rule for training multi-layer feed-forward neural networks with non-linear units, which is commonly known as the *Backpropagation learning rule*.

2.6.1 Generalization of the Delta Rule

As mentioned above, the Backpropagation learning rule is a generalization of the delta rule for training multi-layer feed-forward neural networks with non-linear units. Figure 4 shows such a network.

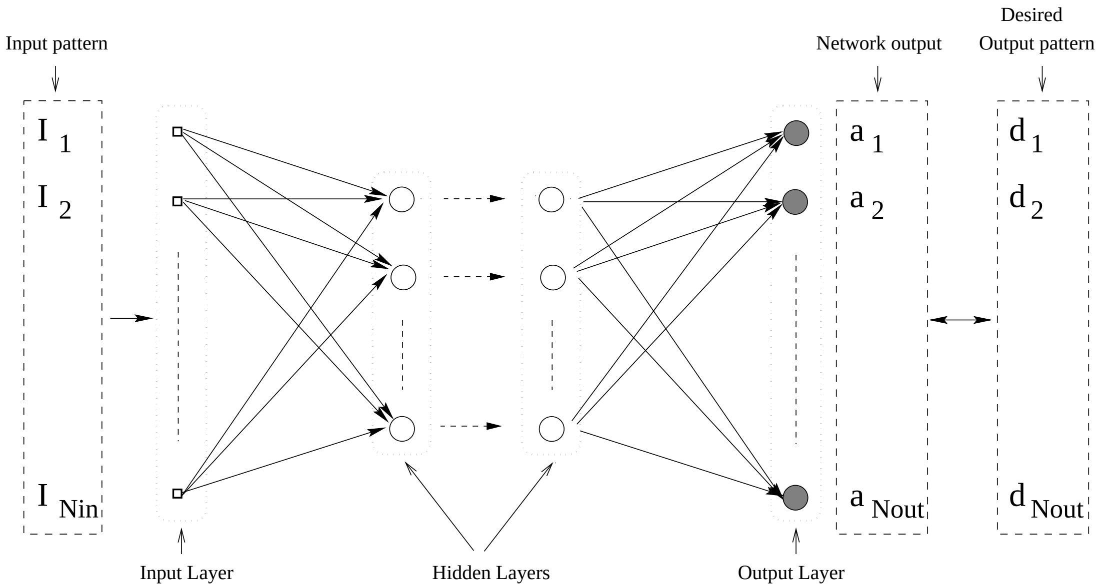

Figure 4: Supervised learning in a multi-layer feed-forward neural network.

In the network, there is an input layer, an output layer, and one or more hidden layers in between. The learning paradigm is supervised learning. During training, the network is given an input pattern for input to the input layer. Based on the given input pattern, the network will compute the output in the output layer. This network output is then compared with the desired output pattern. The aim of the Backpropagation learning rule is to define a method to adjust the weights of the network such that eventually the network will give output that matches the desired output pattern given any input pattern in the training set.

To derive the generalized delta rule, we start by specifying the activation value (a_{pi}) for the i^{th} unit in any layer when the p^{th} input pattern is presented to the network. That is:

$$a_{pi} = F_i(net_{pi})$$

in which

$$net_{pi} = \sum_{j=1}^{N} w_{ij} a_{pj} + \theta_i$$

where F_i is the activation function of the unit, w_{ij} is the weight of the connection from the j^{th} unit in the previous layer to the unit, N is the number of connections to the unit, a_{pj} is the activation value of the j^{th} unit in the previous layer, and θ_i is the bias input to the unit.

We define the total error (E) summed over all training patterns and all output units as:

$$E = \sum_{p=1}^{M} E_p = \frac{1}{2} \sum_{p=1}^{M} \sum_{i=1}^{N_{out}} (d_{pi} - a_{pi})^2$$

where M is the number of training patterns in the training set, N_{out} is the number of output units, d_{pi} is the desired output value of the i^{th} output unit corresponding to the p^{th} desired output pattern, and a_{pi} is the output value of the i^{th} output unit when the p^{th} input pattern is presented to the network.

Generalizing from the delta rule for steepest descend on the error surface, we set:

$$\Delta_p w_{ij} = -\eta \frac{\partial E_p}{\partial w_{ij}}$$

We then rewrite the partial derivative as follows:

$$\begin{aligned} \frac{\partial E_p}{\partial w_{ij}} &= \frac{\partial E_p}{\partial net_{pi}} \frac{\partial net_{pi}}{\partial w_{ij}} \\ &= \frac{\partial E_p}{\partial net_{pi}} a_{pj} \end{aligned}$$

If we let $\delta_{pi} = -\frac{\partial E_p}{\partial net_{pi}}$, we get a weight update rule with equivalent form as the delta rule:

$$\Delta_p w_{ij} = \eta \delta_{pi} a_{pj}$$

The problem then is to derive δ_{pi} for each unit i of the network. To compute δ_{pi}, we start by breaking it down into two factors:

$$\begin{aligned}
\delta_{pi} &= -\frac{\partial E_p}{\partial net_{pi}} \\
&= -\frac{\partial E_p}{\partial a_{pi}} \frac{\partial a_{pi}}{\partial net_{pi}} \\
&= -\frac{\partial E_p}{\partial a_{pi}} F_i'(net_{pi})
\end{aligned}$$

where $F_i'(net_{pi})$ is the value of the derivative of the activation function evaluated at the net input net_{pi}. The requirement is that the activation function must be a differentiable function. One such function is the sigmoid function.

To compute the first factor $\frac{\partial E_p}{\partial a_{pi}}$, we need to consider two cases. The first case is when the unit i is an output unit of the network. In this case, by the definition of E_p we have:

$$\frac{\partial E_p}{\partial a_{pi}} = -(d_{pi} - a_{pi})$$

Therefore, for any output unit i, we get:

$$\delta_{pi} = (d_{pi} - a_{pi}) F_i'(net_{pi})$$

The other case is when unit i is not an output unit. In this case, we use chain rule to write:

$$\begin{aligned}
\frac{\partial E_p}{\partial a_{pi}} &= \sum_{k=1}^{N_{out}} \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial a_{pi}} \\
&= \sum_{k=1}^{N_{out}} \frac{\partial E_p}{\partial net_{pk}} \frac{\partial}{\partial a_{pi}} \sum_{j=1}^{N} (w_{kj} a_{pj} + \theta_k)
\end{aligned}$$

$$= \sum_{k=1}^{N_{out}} \frac{\partial E_p}{\partial net_{pk}} w_{ki}$$

$$= -\sum_{k=1}^{N_{out}} \delta_{pk} w_{ki}$$

and by substitution, we have:

$$\delta_{pi} = F_i'(net_{pi}) \sum_{k=1}^{N_{out}} \delta_{pk} w_{ki}$$

2.6.2 Two Phases of Signal Propagation

From the generalization of the delta rule, we observe that there should be two phases of signal propagation in the network: In the first phase, the input pattern is presented to the network. Activation values from each unit are propagated as signals along the forward direction of the connections from the input layer, via the hidden layers, until the output of the network is computed in the output layer as the activation values of the output units. This output is then compared with the desired output pattern to compute the error signal δ for each of the output units. In the second phase, these error signals are propagated in the backward direction from the output layer, via the hidden layers and eventually to the input layer. During this phase the error signals for all non-output units are computed recursively. After all the error signals are found, the weight changes can then be computed and the weights updated accordingly. During training, this process is done repeatedly to update the weights until the total error (E) falls below some tolerance level.

2.6.3 Generalization and Testing

As mentioned previously, the purpose of training is to find a set of weights such that the neural network will map the set of training input patterns to the corresponding set of desired output patterns correctly. However, it is not useful if the neural network is able to do only just that, as a simple algorithm can do the same by looking up a two-column table with one column for the input patterns and the other column for the corresponding desired output patterns.

The real power of neural networks trained by Backpropagation is their *generalization* ability. Generalization refers to the ability of the neural network to interpolate and give reasonable output when a new input pattern previously unseen by the network during training is presented to it. In order to ensure good generalization, a trained neural network is usually tested by a test set. Like the training examples, the test examples in the test set are input-output pairs, except that these examples are previously unseen by the neural network. For testing, the test input patterns are presented to the neural network as input, and the output given by the network is compared to the set of desired output in the test set to assess the generalization ability of the network.

CHAPTER 3

ARTIFICIAL NEURAL NETWORKS AND INFORMATION RETRIEVAL

In recent years, more and more researchers have proposed the application of artificial neural networks for information retrieval (IR) tasks. Due to the diversified background among different researchers, there are considerable differences between the approaches taken. In this chapter, we will provide a summary of some of the proposals. An attempt will be made to classify the different approaches into two main classes. We will discuss the advantages and disadvantages of the two classes of approaches.

3.1 Classification of the Different Approaches

Even though there are wide variations between the different approaches, the neural network models proposed for information retrieval tasks can be broadly divided into two main classes. The first class of neural network models is mostly domain independent models used in many different domains. These include the Adaline and networks trained by Backpropagation. These models are not specifically designed for the information retrieval tasks. During training, the weights of the network connections are initialized to random values, and the networks learn to perform the particular IR task by cycling through the set of training examples provided and adjusting the connection weights accordingly.

On the other hand, the second class of neural networks is mostly ad hoc approaches designed specifically for solving information retrieval problems and usually differs considerably from domain independent neural network models. In most of these networks, the network units correspond to typical information retrieval objects, such as documents, queries, and indexing terms. Connections exist between different units with weights indicating how close the objects are related to each other. The most important characteristics which distinguishes these neural networks from the first class of networks is that the connection weights are initialized based on some domain knowledge gathered from more traditional IR techniques. We refer to this method of weight initialization based on pre-existing domain knowledge as *pre-programming*. In the following two sections, we will give examples of these two classes of approaches.

3.2 Neural Networks without Pre-programming

Crestani [9, 10] used a 3-layer feed-forward neural network trained by Backpropagation (see section 2.6) for ranked text retrieval. In the proposed system, both documents and queries were represented as binary term vectors. A one in a vector component indicated the presence of a term in the document or query being represented, and a zero indicated the term was absent. For training the neural network, training examples were created from the standard judgements of a test collection (see section 6.1) based on different learning strategies.

In one strategy, a training example consisted of an input-output pair formed by a query vector as the input pattern and a document vector corresponding to a document relevant to the query as the desired output pattern. The problem with this strategy was that the neural network was confused by the fact that a single input pattern (query) was associated with more than one desired output patterns (relevant documents) in the training set. This problem was solved in another strategy by forming a "centroid" cluster representative vector from the set of documents relevant to a query. A training example was then formed with a query vector as input and the corresponding "centroid" vector as the desired output.

After the network was trained, retrieval was done by feeding the term vector of a user input query as input to the network. Based on the input, the network gave an output vector. This output vector was then compared with the set of document vectors corresponding to the documents to be retrieved, and a similarity score was computed based on the Dice's coefficient [12]. The documents were then ranked based on their scores.

Experiments were conducted using a subset of the ASLIB Cranfield test collection [7] for training and testing. The subset used consisted of 200 documents and 42 queries. The performance of the neural network was compared with a conventional information retrieval system based on the Dice's coefficient to measure the similarities between the documents and queries. The results showed that the neural network performed worse than the conventional information retrieval system.

Meghabghab [37] described an application of a neural network for multiversion information retrieval. Multiversion information retrieval (MIR) refers to the situation where redundancy is introduced to an information retrieval system such that more than one algorithm are available for realizing one information retrieval task on hand. As each algorithm may vary in performance depending on the particular user's search strategies and the type of input queries, every time when a retrieval request is received from the user, an MIR system has to choose the best algorithm among the available ones based on the given situation.

In the report, a 3-layer feed-forward neural network trained by Backpropagation was proposed as the algorithm selector for relevance feedback (see section 1.1.1) in an MIR system. The function of the neural network was to select the best feedback algorithm for selecting a set of keywords from the set of documents marked as relevant by the user to augment the query. The different algorithms were compared two at a time. The input to the network was the output of the two algorithms to be compared, which were the two augmented queries created by the algorithms. The network had a single binary output indicating which of the two algorithms being compared was better. To train the network by supervised learning, training examples were created by human experts who compared the different algorithms and indicated which one was better for a set of queries.

The experimental collection consisted of 2500 documents randomly selected from the journal Computer Science VINITI (journal of abstracts), 1982-1987. In the experiment, 20 queries were used for training and 15 queries were used for testing. MIR systems using the neural network and the Dempster-Shafer's model [54, 55] were compared based on their performance on choosing among two query reformulation algorithms for relevance feedback. The results showed that the neural network performed better than the Dempster-Shafer's model.

Lewis *et al.* [33] studied the Adaline (see section 2.5.3) as the classifier model for text categorization. Three different training methods were compared, namely the Rocchio algorithm [50], the Widrow-Hoff algorithm or the delta rule [61], and the Kivinen and Warmuth's EG algorithm [24] which is an extension to the delta rule. Batch training was used for the Rocchio Algorithm to update the weights of the Adaline by taking into account the whole training set at once, while online training was used for the other two algorithms to perform weight update by running through the training examples one at a time.

Experiments were carried out using the OHSUMED test collection [17] and the AP newswire subset of the TREC/TIPSTER test collection [16] for training and testing. The results showed that both the delta rule and the EG algorithm performed significantly better than the Rocchio algorithm. The results also showed that categorization effectiveness was improved by using the $TF \times IDF$ measure (see section 5.3.4) as the term weight instead of a binary term weight in the document feature vectors.

Schütze *et al.* [53] compared a number of classifier models for document routing on the TREC/TIPSTER test collection. Document routing is a text filtering task in which documents are classified as being either relevant or non-relevant to a routing query. In the report, five different methods of generating classifiers for text filtering were compared, namely Rocchio algorithm [50], linear discriminant analysis (LDA), logistic regression [13], linear neural networks and non-linear neural networks trained by Backpropagation. The proposed "linear" neural network model was different from the Adaline in that they used the non-linear sigmoid function as the activation functions for all network units. The difference between the "linear" and non-linear neural network models lied only in the absence and presence of a hidden layer.

The high dimensionality of the initial feature space of textual data was identified as a major hindrance for applying standard classification techniques for text filtering. Two different methods of dimensionality reduction (see chapter 4) were compared: Latent Semantic Index (LSI) [11] and term selection (see section 5.1.3). In the first method, separate LSI was done for each query, each focusing on the set of documents closest to the particular query according to some distance measure. In the second method, terms were ranked based on an importance measure. The importance score of a term was computed by considering 4 term occurrence statistics: the number of relevant and non-relevant documents in which the term occurred and the number of relevant and non-relevant documents in which the term did not occur.

Their results showed that query expansion was the worst in performance among the classification methods compared. Logistic regression was less effective than LDA and neural networks, although when LSI was used for dimensionality reduction the difference in performance was small. Performance of LDA and neural networks was about the same when LSI was used. However, neural networks were significantly better than LDA when term selection was used for dimensionality reduction. There were no significant difference in performance between the linear and non-linear neural networks regardless of the dimensionality reduction technique used.

Mock and Vemuri [39] described a neural network for USENET news filtering. In their system, the user was required to provide feedback to the system by indicating if the messages he read were relevant or not relevant to his needs. This feedback information was then used as training examples to train the neural network by supervised learning. The trained network was used to predict relevance for unread articles. In their discussion, they pointed out that traditional neural networks trained by Backpropagation using only gradient descend was not suitable for this situation. The number of training examples and training time required for reasonable training were too large to be acceptable to users who want the system to learn and adapt quickly to their interests. For this reason, they proposed to use a special type of radial basis function neural network [42].

The idea was to combine gradient descent with memorization in the same neural network. When the network poorly classified an input article, memorization was done by creating a new hidden unit for matching the input correctly. This new hidden unit was created with input weights matching the input vector corresponding to the input article, and output weights were set to the desired output. On the other hand, if the output of the network was close to the desired output, gradient descend was applied to adjust the weights even closer to the desired output. When the network was applied for filtering the articles, memorization was done initially and when new cases were encountered, while gradient descent was used for fine-tuning the network.

Experiments were conducted using a set of 100 messages extracted from the "comp.ai" artificial intelligence USENET newsgroup. Half of the messages were used for training and the other half used for testing. Three filtering models were compared: global hill climbing, local hill climbing using genetic algorithm [2], and artificial neural network. In the experiments, global hill climbing gave the highest accuracy, followed by local hill climbing, and the neural network performed slightly worse than the other two models.

3.3 Neural Networks with Pre-programming

Kwok [27, 28] proposed a three-layer neural network implementation of probabilistic information retrieval. The network consisted of a document layer, a term layer, and a query layer, with the term layer being the hidden layer in between the other two layers. Units in the three layers corresponded to the documents, index terms and the queries respectively. There were bidirectional connections with asymmetrical weights between the document layer and the term layer, and between the term layer and the query layer. The weight of a connection from a document (query) unit to a term unit was set to the probability of the term being used in the document (query). On the other hand, the weight of a connection from a term unit to a document (query) unit was the probabilistic weight of the term in the document (query). This was estimated based on the conditional probabilities of the term's presence given relevance or non-relevance.

Retrieval was done by focusing on a particular query. The documents were ranked based on their relevance to that query. Alternatively, by focusing on a document, the queries can be ranked based on their relevance to that document. In a query focused retrieval, activation of each document unit was clamped to 1 in turn. These activations were spread to term units and then to the query unit in focus through connections with non-zero weights. The activation received at the query unit in focus was then the sum over all terms in common between the document and the query, the product of the probability of term usage in the document (weight of the document-term connection) and the probabilistic term weight in the query (weight of the term-query connection). A document was considered relevant only if the activation received at the query unit in focus exceeded some threshold. A document focused retrieval was done in a similar fashion but with activations spreading in the opposite direction. In the proposal, a learning rule was also provided to adjust the connection weights based on information received from relevance feedback.

Two sets of experiments were performed. In the first set of experiments, four collections were used: MED, CRAN, CISI and CACM (see table 1 for collection statistics). In the second set of experiments, Wall Street Journal news articles from the TREC/TIPSTER collection were used. The results showed that the learning mechanisms for relevance feedback were effective in improving retrieval effectiveness. However, the effect was less significant in the second set of experiments than in the first. One explanation was that better queries were formed from the TREC/TIPSTER collection using the long topic descriptions, so that the effect of query expansion on retrieval effectiveness was reduced.

Wilkinson and Hingston [18, 62] proposed to implement the cosine similarity measure of the vector space model [50] in a three-layer neural network for ranked text retrieval. The proposed neural network architecture was similar to that used by Kwok [27, 28]. The network had a query layer, a document layer and a term layer in between the other two layers. Units in the layers corresponded to the user input query, the documents, and the index terms respectively. A bidirectional connection with symmetric weights existed between a document unit and each of the term units corresponding to the terms contained in the document. When the user input a query, a query unit was created in the query layer, with a unidirectional connection from the query unit to each of the term units in the term layer corresponding to the query terms. To initialize the connection weights, two factors ($f1$ and $f2$) were defined based on the cosine similarity measure:

$$f1_{ij} = \frac{d_{ij}}{(\sum_{j=1}^{t} d_{ij}^2)^{1/2}}$$

$$f2_j = \frac{q_j}{(\sum_{j=1}^{t} q_j^2)^{1/2}}$$

where t was the number of terms in the vocabulary, d_{ij} was the weight $(TF \times IDF)$ of term j in document i, and q_j was the weight of term j in the query.

The weight of a document-term connection between document unit i and term unit j was set to $f1_{ij}$, and the weight of a query-term connection between the query unit and term unit j was set to $f2_j$. Any unit was said to be activated only if its activation exceeded a certain threshold. To start a retrieval, the query unit was given an activation of 1. This activation was spread through the query-term connections and then the term-document connections to reach the document units. During this process, only term units corresponding to terms in common between the query and the receiving document units were involved. In this case, the activation received by each document unit was equal to the product $f1_{ij} \times f2_j$ summed over all terms in common between the query and the document, which was the standard document score based on the cosine similarity measure.

After the initial activations of the document units were computed, activations then spread from the activated document units back to the term units such that term units corresponding to terms contained in high score documents were activated. The activation continued to spread from the term units back to the document units with the aim of activating other document units with terms in common with the initially activated document units. This can be viewed as a kind of built-in thesaurus, by which terms in common between documents were used as the bridge to retrieve more documents based on the initial set of retrieved documents. Relevance feedback was implemented by adding a bias input to each document unit, such that if the document was marked as relevant by the user, a positive bias was given to increase its activation. Otherwise, a negative bias was given to dampen its activation.

The neural network was tested using the CACM test collection. The results showed that spread activation was able to give slight improvements in retrieval effectiveness over the pure vector space model initially. However, retrieval effectiveness dropped as more units were activated in later iterations, which resulted in random ordering.

Chen *et al.* [5, 4, 6] proposed to use the Hopfield network's parallel relaxation mechanism [20] to identify concepts (terms) related to an initial set of input concepts. The Hopfield network was a single layer fully connected recurrent neural network. Each unit in the network corresponded to a unique term, the connection weights between units corresponded to the similarities between pairs of terms. The weights were set based on the cosine similarity measure or based on external thesauri through a user-guided procedure [3].

At start, a set of terms representing the initial concepts was provided as the input pattern to the network, such that corresponding units were activated (activation set to 1). Parallel relaxation was then used to activate more units with relatively high similarity (strong weight) to the input terms. This process was repeated until the network became stable such that no more new units were activated. The set of activated units after the process represented the set of identified concepts.

The neural network model was used to identify concepts from a set of brainstorming comments generated during an electronic meeting [4]. The results given by the neural network were compared with those given by four human subjects. Experimental results showed that although two of the human subjects out-performed the neural network in precision and recall, the performance of the neural network was not statistically different from those of the other two human subjects.

Syu and Yang [57] proposed to use a two layer feed-forward neural network to implement a Bayesian inference network based on the parsimonious covering theory [44] for text retrieval. The neural network consisted of a document layer and an index term layer. In the document layer, each unit corresponded to a document in the document database. In the index term layer, each unit corresponded to an indexing term in the vocabulary. A weighted connection existed from a document unit to a term unit if the term appeared in the document. A constant weight was set for each document-term connection with value equal to the $TF \times IDF$ score of the term in the document. All units in the network had activation values in the range [0, 1]. The activation values of all units were changed based on spread activation until a stable stage was reached. This process was similar to the parallel relaxation mechanism in Hopfield networks.

To start a retrieval, the activation of each document unit was set to the estimated prior probabilities of the documents ($1/N$, where N was the number of documents in the database). An external thesaurus was used to relate the index terms, such that all units corresponding to the query terms and terms having the same meaning as any query term were marked present. During the spread activation, only term units that were marked present participated. The activations of the document units were updated by following a set of pre-defined updating rules. These rules compute the amount of weight update based on the connection weights and the estimated prior probabilities of the documents. The spread activation continued until a stable stage was reached when activation values of document units converged to either 0 or 1. The set of documents corresponding to units having an activation level near to 1 was retrieved for the input query.

Performance of the inference network was evaluated using the CACM test collection. Retrieval effectiveness of the neural network was compared to that of a Bayesian network [58] based on precision and recall. The results showed that the neural network was able to give performance better (9.7% higher in average precision) than the Bayesian network.

3.4 Comparing the Different Approaches

From the examples given in the last two sections, we can make the following observations about the two different classes of approaches to neural network based information retrieval:

- In the first class of approaches where there is no pre-programming, there must be enough training before the IR systems can give reasonable performance. In these systems, training is of primary importance without which the systems cannot work. In order to speed up learning and to reduce the time needed for the system to reach reasonable performance initially, some kind of memorization ability can be introduced as in the approach proposed in [39].

- In the second class of approaches, pre-programming enables the IR systems to have reasonable performance even without any training. In these systems, training is of secondary importance in that it is used only to fine-tune the systems for better performance when new examples become available. In fact, some of the proposals discussed in section 3.3 did not support any kind of training at all.

- Although pre-programming ensures the IR systems to be usable before they are trained, a considerable amount of human efforts is required to derive the initial set of weights to ensure good performance. These efforts may include consulting with domain experts for the properties of the particular task on hand. On the other hand, for systems without pre-programming, the neural network models can usually be used as a black box. However, human efforts are still required to create a set of training examples for adequate training.

- In both classes of approaches, scalability is a problem as the size of the neural networks usually grows with the number of IR objects existing in the task. In order to improve scalability, one possible solution is to reduce the number of indexing terms by using dimensionality reduction techniques as in the approach proposed in [53].

CHAPTER 4

PRINCIPAL COMPONENT ANALYSIS AND DIMENSIONALITY REDUCTION

In this chapter, we will introduce a statistical multivariate data analysis technique known as *Principal Component Analysis* (PCA), or *Karhunen-Loève transformation*. Although principal component analysis can be applied in many ways, such as for data interpretation or for displaying multivariate data graphically, this chapter is focused on its application on dimensionality reduction of a high dimensional data set.

This chapter serves only as a brief introduction to the ideas of dimensionality reduction by principal component analysis. Interested readers may refer to [22, 23] for thorough discussions on principal component analysis.

4.1 Purpose of Dimensionality Reduction

In many tasks, the input data is usually given as a set of data objects, where each object is represented by n characteristics or variables. This set of data objects can be represented geometrically in an n-dimensional data space with each dimension corresponding to one of the variables. Each object is then a point in this data space with coordinates defined by the values of its variables. In this case, we say that the set of data objects is an n-dimensional data set, and the number of dimensions in the data space is referred to as the *dimensionality*.

Due to the nature of some tasks, the dimensionality of the input data can be large where each object is represented by a large number of variables. For more efficient processing, it is often necessary to transform the original data set such that each object is represented by a smaller number of variables. This is equivalent to compressing the original n-dimensional data space into a new reduced data space of p ($< n$) dimensions, where each data object is represented by p (possibly new) variables. We refer to this process of reducing the dimensionality of the original data space as *dimensionality reduction*.

As we reduce the dimensionality of the original data space, it is inevitable that some of the intrinsic information content present in the original input data is lost. Our goal in dimensionality reduction is thus to retain as much of the information content in the original data as possible.

4.2 The Idea of Principal Components

In principal component analysis, the amount of information content in the data is measured by the data variance. The main idea of principal component analysis is to reduce the dimensionality of the original data set while retaining the largest amount of the variance intrinsic in the original data.

As each data object is represented by a set of variables, we can view the variance in the data as being distributed among these variables. Usually, there are some correlations between the variables. In principal component analysis, we try to take advantage of these correlations and linearly transform the original set of variables into a new set of uncorrelated variables. As a result of this transformation, there will be redistribution of variance among the variables such that the new uncorrelated variables will have decreasing variance, with the first variable having the largest variance and the last variable having the least variance. After this variance redistribution, we can reduce the dimensionality of the original data by dropping the dimensions corresponding to the variables with small variance, while retaining the ones with large variance. In this way, we can ensure the largest amount of variance is retained. In the following, we will describe this process in details.

4.2.1 Principal Component Analysis

Suppose we are given an n-dimensional data set, we can represent this data set by an n-dimensional random vector ($\mathbf{x}$):

$$\mathbf{x} = < x_1, x_2, \ldots, x_n >$$

where the i^{th} random variable in $\mathbf{x}$ (x_i) takes on values from the i^{th} variable in the data objects. In order for principal component analysis to proceed on the given data set, we assume that $\mathbf{x}$ has zero mean [1], that is:

$$E[\mathbf{x}] = 0$$

The data *covariance matrix* ($\mathbf{R}$) is then defined as:

$$\mathbf{R} = E[\mathbf{x}\mathbf{x}^T]$$

which is an n by n symmetric matrix calculated as the expectation of the outer product of the vector $\mathbf{x}$ with itself.

We now find a set of n n-dimensional unit vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$, such that these vectors are orthogonal to each other:

$$\mathbf{u}_i^T \mathbf{u}_j = \begin{cases} 1, & j = i \\ 0, & j \neq i \end{cases}$$

This set of n unit vectors represents an orthonormal basis for the n-dimensional data space. We form projections of $\mathbf{x}$ (a_i) onto the set of unit vectors:

$$a_i = \mathbf{x}^T \mathbf{u}_i$$

In doing so, we perform a coordinate transformation in the data space, such that the unit vectors ($\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$) form the axes of the new coordinate system.

[1] If this is not the case, $\mathbf{x}$ can be transformed into such by subtracting the mean from it.

In this way, the original data vector ($\mathbf{x}$) is transformed into a new random vector ($\mathbf{a}$) with respect to the new coordinate system:

$$\mathbf{a} = < a_1, a_2, \ldots, a_n >$$

The projections (a_i) represent a set of new variables for the data objects created by the transformation.

In principal component analysis, the choice of the unit vectors ($\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$) is such that the projections (a_i) are uncorrelated with each other. Moreover, if we denote the variance of a_i by λ_i, for $i = 1, 2, \ldots, n$, then the following condition is satisfied:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_n$$

In other words, the new variables (a_i) contain decreasing variance, such that the first variable (a_1) has the largest variance, followed by the second variable (a_2) and so on, with the last variable (a_n) having the smallest variance. These new variables (a_i) are called *principal components*. Figure 5 illustrates this for a 2-dimensional data set.

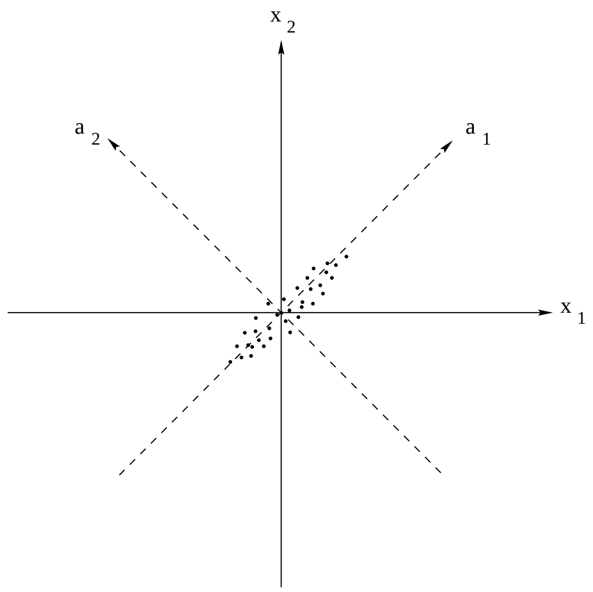

Figure 5: Principal components in 2-dimensional data space.

It can be shown that the variance $(\lambda_1, \lambda_2, \ldots, \lambda_n)$ corresponds to the eigenvalues of the data covariance matrix $(\mathbf{R})$ arranged in descending order, and the unit vectors $(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n)$ are the corresponding eigenvectors of the data covariance matrix $(\mathbf{R})$ [23].

4.2.2 Dimensionality Reduction

For the purpose of dimensionality reduction, we need to reduce the dimensionality of the data space from n to p, where $p < n$. This is done by discarding those variables with small variance and retaining only those having large variance. Thus, we form a reduced random vector $(\mathbf{a}_p)$ with p-dimensions from the n-dimensional random vector $\mathbf{a}$ by taking only the first p variables:

$$\mathbf{a}_p =< a_1, a_2, \ldots, a_p >$$

In this way, the dimensionality of the data space is reduced from n to p with minimal loss in data variance. In the reduced data space, each data object is represented by p variables corresponding to the first p principal components.

The portion of the data variance represented by the first p principal components is given by:

$$\frac{\sum_{i=1}^{p} \lambda_i}{\sum_{j=1}^{n} \lambda_j}$$

4.3 Method of Principal Component Analysis

In this section, we will describe the practical issue of finding the principal components from a given data set.

In order to perform principal component analysis, we represent an n-dimensional data set with m data objects by an m by n matrix $(\mathbf{D})$:

$$\mathbf{D} = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{pmatrix}$$

where d_{ij} is the value of the j^{th} variable in the i^{th} data object. For the analysis to proceed, we require that the mean of the elements in each column of $\mathbf{D}$ be zero, that is:

$$\bar{d}_j = \frac{\sum_{i=1}^{m} d_{ij}}{m} = 0 \qquad \text{for } j = 1, 2, \ldots, n$$

This zero mean property is ensured by subtracting the column mean ($\bar{d}_j$) from each corresponding column element of $\mathbf{D}$, producing an m by n matrix ($\mathbf{X}$) with zero mean:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}$$

where $x_{ij} = d_{ij} - \bar{d}_j$, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

To obtain the principal components, we must compute the eigenvectors of the data covariance matrix. A straightforward way is to obtain the n by n data covariance matrix and compute the desired eigenvectors using one of the methods for computing the eigvenvalues and eigvenvectors of a symmetric matrix.

An alternate method is to obtain the *singular value decomposition* (SVD) of the matrix $\mathbf{X}$ [22]:

$$\mathbf{X} = \mathbf{USV}^T$$

The matrix $\mathbf{S}$ is an m by n "diagonal" rectangular matrix, with diagonal elements $(s_{11}, s_{22}, \ldots, s_{nn})$ arranged in descending order:

$$s_{11} > s_{22} > \ldots > s_{nn}$$

The diagonal elements of $\mathbf{S}$ correspond to the square roots of the eigenvalues of the covariance matrix [2], that is:

$$s_{ii} = \sqrt{\lambda_i} \qquad \text{for } i = 1, 2, \ldots, n$$

The matrix $\mathbf{V}$ is n by n, with the i^{th} column corresponding to the i^{th} eigenvector of the covariance matrix:

$$\mathbf{V} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n]$$

The m by m matrix $\mathbf{U}$ is not useful for our purpose and is simply discarded. The advantage of this method is that the desired eigenvectors can be obtained in one step without obtaining the data covariance matrix first.

We can then obtain the first p principal components by multiplying $\mathbf{X}$ with the matrix $\mathbf{V}_p$ formed by the first p columns of $\mathbf{V}$:

$$\mathbf{A} = \mathbf{X}\mathbf{V}_p$$

where $\mathbf{V}_p = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p]$.

The matrix $\mathbf{A}$ is an m by p matrix of the form:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mp} \end{pmatrix}$$

where a_{ij} corresponds to the j^{th} principal component of the i^{th} data object.

[2] We assume $m > n$ here. In the case when $n > m$, the covariance matrix has at most m non-zero eigenvalues corresponding to the m diagonal elements of $\mathbf{S}$. In this case, we only need to obtain the eigenvectors with non-zero eigenvalues (see section 2.4 of [23]).

CHAPTER 5

TEXT CATEGORIZATION AS A LEARNED CLASSIFICATION PROBLEM

In this chapter, we will present a model for text categorization based on a non-linear feed-forward neural network trained by Backpropagation learning rule. In the proposed model, we formulate the text categorization problem as a classification problem and the neural network as a classifier created by supervised learning. In order to improve the scalability of the model, various dimensionality reduction techniques have been proposed to reduce the high dimensionality of the raw data before feeding them to the neural network. We will describe these techniques in detail in this chapter.

5.1 Classification Problems in General

Given a set of objects, it is often desirable to divide the objects into a number of different groups. The grouping of objects is usually based on the characteristics or properties that the objects possess. For example, given a bag of fruits, it is often desirable to divide the fruits into groups such as apples, oranges and bananas. In this case, the characteristics based on which the fruits are separated into groups may be the shapes and colors of the fruits. The task of grouping objects into different groups based on their characteristics or properties is well studied and is often known as classification. The groups into which the objects are classified are called classes.

5.1.1 Differences between Classification and Clustering

It is important to distinguish between two related but different tasks, namely classification and clustering. In classification, the classes are assumed to be pre-defined. The job of the classification module, or the classifier as it is often being called, is to identify a subset of the pre-defined classes to be assigned to the object. On the other hand, in the clustering task, the classes are often not known in advance. The clustering module is then required to discover appropriate classes and group the objects accordingly. The criterion for class discovery is usually based on some similarity measure quantifying how similar or how closely related the objects are to each other. Objects that are closely related are usually grouped to form a cluster, and objects that are not similar to each other are put into separate clusters. Thus, as is the case in many clustering techniques, the goal of clustering is to form clusters of objects such that the intra-cluster object similarity is maximized, but the inter-cluster object similarity is minimized.

5.1.2 Computer-Based Classification

As mentioned previously, the decision of classifying objects into a number of pre-defined classes is based on the characteristics or properties of the objects. In order for a computer to carry out the classification task, an effective representation of the objects should include all the properties that are useful for identifying the class or classes to which the objects should be assigned.

In computer-based classification, an object is usually represented by a set of attributes, or features. Each feature corresponds to a property of the object being represented. Usually, the set of features is grouped to form a feature vector, in which each vector component corresponds to a feature. For example, the following is a feature vector (X) with n features ($f_1, f_2, \ldots, f_n$):

$$X =< f_1, f_2, \ldots, f_n >$$

The data type of the feature vector components depends on the type of features being represented.

Figure 6 shows a general model of the computer based-classification task. In this model, a set of n-dimensional feature vectors representing the objects to be classified is given to the classifier as input. Based on the features contained in

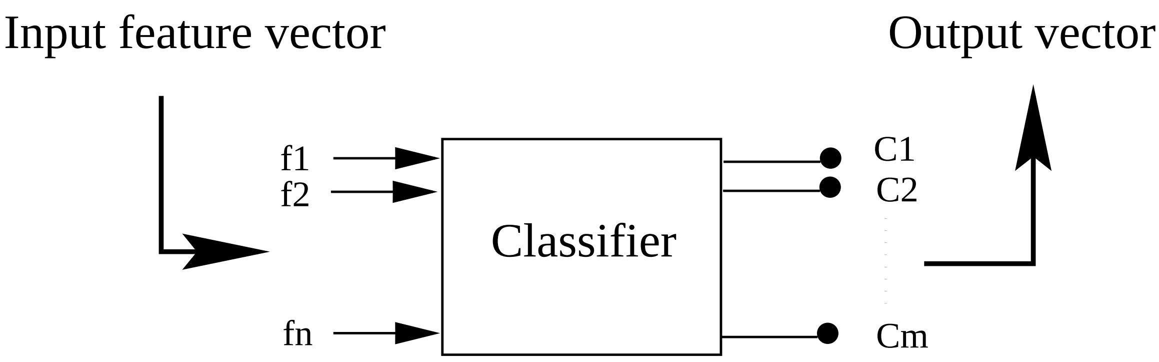

Figure 6: A general model of computer based classification.

each input feature vector, the classifier will make the decision of class assignment for the objects. This decision is represented by an m-dimensional output vector, where m is equal to the number of pre-defined classes in the particular classification task. In this case, the classifier can be seen as a function which maps an n-dimensional feature vector into an m-dimensional output vector. The output vector from the classifier is sometimes called a classification vector [32].

Depending on the classification task, the output vector can be either a bit vector or a real-valued vector. In binary classification, class membership of an object is binary. That is, a class is either assigned or not assigned to an object. In this case, the output vector is a binary vector. A one in the i^{th} component of the output vector indicates that class i is assigned to the object, while a zero means that class i is not assigned to the object. If there are overlaps between classes, where an object can be assigned more than one class, two or more bits in the output vector will be set to 1.

On the other hand, in graded or probabilistic classification, the output vector will be a real-valued vector, where each output vector component is a real number, usually in the range [0, 1]. The i^{th} vector component is a real number representing the probability or likeliness of the object belonging to class i. A greater value in the i^{th} vector component indicates higher probability or likeliness of the object being a member of class i, while a smaller value means the opposite.

5.1.3 Feature Selection and Feature Extraction

It has long been recognized that the set of features used to describe the objects to be classified can have a great impact on the accuracy of the classification system [32]. Therefore, in order to improve the accuracy of the classifier, coming up with a set of "good" features for object representation is sometimes the most time consuming and critical step in the construction of a classification system. In many cases, this involves first identifying a set of "raw" features or measurements of the various object properties, and then refining on this set of "raw" features in order to come up with another set of "improved" features more suitable for the particular classification task.

In the literature, this process of feature refinement is often known as feature selection or feature extraction. While many authors have used the two terms interchangeably, here we choose to follow the definitions used in [32] and distinguish these as two different ways of refining an initial feature set for improved object representation.

Feature Selection

Feature selection is the process of selecting a subset of d features from an initial set of D features, where $d < D$. In many cases, this is done manually in such a way that some human experts choose those features from the initial set that they think are useful for the classification task on hand and drop those that are not. The problem with manual approach is that it is time consuming and requires a lot of experience and domain knowledge to choose a high quality feature set. As a result, it is desirable to automate the feature selection process.

In order to automate the feature selection process, we must have some scoring scheme for evaluating the quality of each feature in the initial set. Automated feature selection can then be done by ranking the individual features from the initial set according to the scoring scheme chosen, and selecting only a subset of those features having a higher score than others.

It should be pointed out that as feature selection will always result in a reduced number of features, it can be considered a technique for dimensionality reduction.

Feature Extraction

One of the problems with feature selection is that there is always information loss as some of the low score features are being filtered out. Although a good scoring scheme can minimize the adverse effect of information loss on classification accuracy by ensuring that the features being filtered out are insignificant to the classification task, this is rarely optimal especially when the features are not independent to each other. Moreover, feature selection is not effective if none of the original features are good, as there are no new features being created. For this reason, it is sometimes desirable to apply feature extraction for the creation of new features based on the original feature set.

In [32], Lewis described the feature extraction process as the application of a set of operators on one or more original features in which one or more new features are produced. Depending on the number of features being acted on and the number of features being produced, four types of operators can be distinguished: one-to-one operators which transform a single feature into a single new feature, one-to-many operators which take a single feature and produce multiple new features, many-to-one operators which take multiple features and produce a single new feature, and many-to-many operators which take multiple features and produce multiple new features. According to the above definitions, principal component analysis can be regarded as a many-to-many operator for feature extraction, in which a number of features are transformed into a set of new features called the principal components (see section 5.4.4 for more details).

5.2 Text Categorization as a Classification Problem

As mentioned in section 1.1.3, text categorization can be described as a task in which a set of text documents is to be classified into a set of pre-defined categories. It is therefore natural for us to formulate the text categorization task

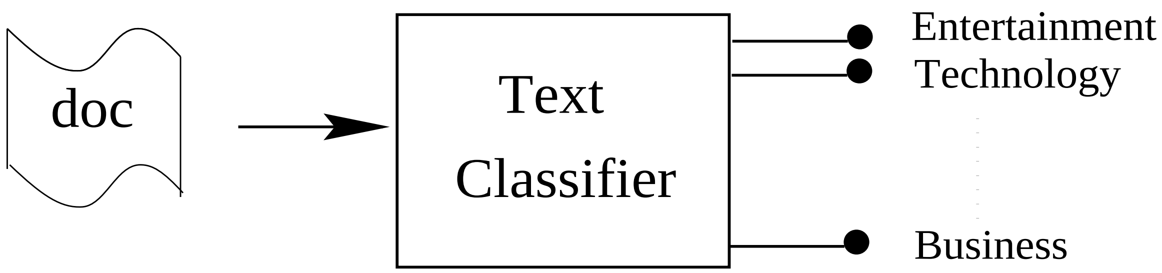

Figure 7: A classification model for text categorization.

as a classification problem. In this case, borrowing from the terminology used in section 5.1, the set of input objects to the classifier is the set of text documents being categorized, and the set of pre-defined categories corresponds to the set of pre-defined classes into which the objects are classified. Figure 7 illustrates the basic classification model.

In order to build a text categorization system according to the general classification model given in figure 6, the set of documents must be transformed into a set of feature vectors. This transformation requires two steps. The first step is to identify an initial feature set based on the text content of the documents. This step is done by applying a set of text processing techniques commonly used in automatic text indexing systems. Some of these techniques can be considered as a set of feature selection and feature extraction techniques specific for textual data.

Due to the inherent properties of textual data, there will be a large number of initial features identified after the first step. In order to improve the scalability of the text classification system, dimensionality reduction techniques are needed in the second step to produce a more compact feature set with reduced number of features. This compact set of features will then be fed to the text classifier as the input feature vectors. We will discuss these two steps in more details in the next two sections.

For classification, we propose to use a three layer feed-forward neural network trained by Backpropagation as the text classifier. As the learning paradigm for Backpropagation neural networks is supervised learning, the training set should include a set of training documents, together with a specification of which of the pre-defined categories each document belongs to. The neural network based text classifier will be discussed in detail in section 5.5.

5.3 Identifying Features from Text Documents

In order to identify a set of initial features from text documents, a number of text processing techniques is employed. In this section, we will describe these techniques in more detail. The order of description will follow the order in which the techniques are applied for feature identification in text documents.

5.3.1 Word Extraction

A text document can be viewed as a long stream of characters. For the purpose of feature identification, it is necessary to convert a text document from a long stream of characters into a stream of words or tokens, where a word or token is defined as a group of characters with collective significance [12]. In information retrieval, this text tokenization process is often called word extraction, word breaking, word segmentation, or lexical analysis.

Depending on the natural language the text document is written in, the word extraction process can involve very different techniques. This process is relatively easy in languages such as English, where boundaries between words are marked by special word-delimiting characters such as spaces and punctuations. On the other hand, word extraction is particularly difficult for languages such as Chinese, Japanese, and Korean, where the orthography does not use spaces between words. In order to identify the word boundaries between individual words in these languages, more sophisticated techniques must be applied. In this thesis, we concern ourselves only with English documents. As such, the discussions below will be based on techniques applicable to English documents.

In designing the word extractor, we must define what constitutes a word in the operational point of view. That is, which kind of character sequences should be considered as an English word. This decision varies between different word extractor designs, especially on the handling of special characters such as digits and hyphenations.

After word extraction, a document is transformed into a stream of words with the order of word occurrence in the document preserved. However, in our design, as in many other systems, the ordering information of word occurrence is ignored. In this case, the stream of words from word extraction can be treated as an unordered set of words, with the word ordering or positional information discarded.

5.3.2 Stop Words Removal

It is well recognized among the information retrieval community that a set of functional English words (such as "the", "a", "and", "that", etc) is useless as indexing terms. Salton and McGill [52] described these words as having very low discrimination value, since they occur in almost every English document, and therefore do not help in distinguishing between documents with contents that are about different topics. For this reason, these words are not useful in the text categorization task and thus should be removed from the set of words produced by word extraction.

The process of removing the set of non-content-bearing functional words from the set of words produced by word extraction is called stop words removal, and the functional words being removed are called stop words. In order to remove the stop words, a semi-automatic procedure is followed. This involves first creating a list of stop words to be removed, which is sometimes called the stoplist or a negative dictionary. After this, the set of words produced by word extraction is then scanned so that every word appearing in the stoplist is removed.

The major difficulty in stop words removal is in deciding which words should be put into the stoplist. This is difficult as the list of stop words is sometimes dependent on the topics of the given documents. Words that are not normally considered as stop words in general English may be considered as such in specialized document collections. For example, words such as "home", "page", "world", "web" which are not considered as stop words in general English may be considered as such in a collection of World Wide Web related documents. In the case where the topics of the documents are unknown during the construction of the system, a stoplist for general English documents is usually employed.

It should be noted that according to the definitions in section 5.1.3, stop words removal can be considered a feature selection method where the selection decision is done manually in the creation of the stoplist.

5.3.3 Word Stemming

In a text document, a word may exist in a number of morphological variants. For example, the word "compute" may also exist in its other morphological variants such as "computing", "computed", "computational", or "computer". While these morphological variants are different word forms, they represent the same concept. For information retrieval tasks, including text categorization, it is generally desirable to combine these morphological variants of the same word into one canonical form. In information retrieval, the process of combining or fusing together different morphological variants of the same word into a single canonical form is called stemming, or conflation. The canonical form produced by stemming is called a stem. The module performing stemming is sometimes called a stemmer.

There are various approaches to stemming. The easiest way is to build a translation table with two columns, corresponding to the original word forms and their stems. Stemming is then carried out by looking up the translation table. In order to improve efficiency, indexing methods such as B-tree or hashing can be employed to build an index of the table entries. However, one obvious drawback of this approach is that the translation table has to be built manually, which requires a lot of efforts and can be time-consuming. Another problem is that it is very difficult to build a translation table extensive enough to include every possible word that may exist in the documents. Some words, especially those that are specialized words in particular domains, are likely to be missed.

Given the drawback of table based stemmers, it is then desirable to build stemmers which produce stems based on a set of pre-defined word translation rules. This approach is commonly used in a class of stemmers called affix removal stemmers, in which the stem is formed by removing suffixes and/or prefixes from the original word forms. Affix removal stemmers can be found in various proposals in the literature [34, 49, 43, 41].

The simplest kind of affix removal stemmers is one that removes plurals by removing the suffix "s". The problem with this simple stemmer is obvious as not all words ending with an "s" are plurals (e.g. success, various, etc) and some plurals do not end with an "s" (e.g. criteria, formula, etc). To avoid this problem, stemmers use rules that are conditional statements which will be applied only if some conditions are met. For example, a set of conditional rules for plural removal may look like this [15]:

```
if a word ends in "ies" but not "eies" or "aies"
    then replace the suffix "ies" by "y"

if a word ends in "es" but not "aes", "ees", or "oes"
    then replace the suffix "es" with "e"

if a word ends in "s", but not "us" or "ss"
    then remove the suffix "s"
```

Typical stemmers consist of different sets of rules for different suffix or prefix translations (e.g. "ed" → NULL, "ational" → "ate", etc). These rules may be divided into a number of steps [43], where a rule in each step is applied successively for multiple translations.

Stemming can be considered as a many-to-one operator used for feature extraction. In the stemmers discussed above, human knowledge is needed in creating the set of translation rules used in stemming.

5.3.4 Term Weighting

After word extraction, stop words removal and stemming, each text document is transformed into a set of stems corresponding to the set of words appearing in each document. The next step is to find the union of all these sets, such that the union set contains the set of stems corresponding to all of the words appearing in the given set of documents. Duplicates are removed so that each stem is unique within the union set. In information retrieval, this set of stems constitutes the set of indexing terms for the set of documents. We call this set of indexing terms the *indexing vocabulary*.

In order to produce the set of initial features from the indexing vocabulary, measurements must be made for each term in the vocabulary according to the importance of the term in the documents. This involves assigning to each term a weight indicating the relative importance of the term in a document. This process of assigning a weight to each indexing term is commonly known as *term-weighting* in information retrieval [51]. With the set of indexing terms in the indexing vocabulary and their corresponding term weights, a document can be represented as the following feature vector:

$$D_j = < w_{j1}, w_{j2}, \ldots, w_{jK} >$$

where K is the number of indexing terms in the vocabulary, or the vocabulary size, and w_{ji} is the term weight of the i^{th} indexing term in document j.

There are many different approaches for determining the term weights. The simplest method involves a binary term weight. That is, term weight (w_{ji}) is equal to 0 if the i^{th} indexing term is not in document j and 1 otherwise. In this case, the feature vector representing a document is a bit vector with each bit corresponding to the term weight of each indexing term for that document. This binary representation for documents has been used quite commonly in learned information retrieval tasks [8, 14].

Another approach makes use of the *term occurrence frequencies* information. In this approach, the assumption is that a term that is mentioned more frequently in a document carries more weight in representing the topic of that document, and thus is more important and should be assigned a higher term weight. In this case, the term weight (w_{ji}) is a positive integer equal to the *term frequency* (TF) of the i^{th} indexing term in document j, which is the number of times that the i^{th} indexing term appears in document j.

The problem with the previous approach is that terms with high term frequencies may not always help in distinguishing between documents with different topics, especially when these terms appear frequently in almost all documents in the document set. In order to reflect the *discrimination value* of each term in the term weight, a factor called *inverse document frequency* (IDF) is introduced. A commonly used definition of the IDF is as follows [56]:

$$IDF_i = log\frac{N}{n}$$

where IDF_i is the inverse document frequency of the i^{th} indexing term, N is the number of documents in the document set, and n is the number of documents in which the i^{th} indexing term appears. By this definition, a term that appears in fewer documents will have a higher IDF. The assumption behind this definition is that terms that are concentrated in a few documents are more helpful in distinguishing between documents with different topics. For term weighting, the term weight is equal to the product of the term frequency (TF) and the inverse document frequency (IDF):

$$w_{ji} = TF_{ji} \times IDF_i$$

In this case, terms that appear frequently in a few documents are given higher term weights. An extensive survey of the various term-weighting approaches used in text retrieval systems is given in [51].

5.4 Dimensionality Reduction for Text Categorization

After a set of features is identified as described in the last section, the set of text documents is transformed into a set of feature vectors with each feature corresponding to the term weight of an indexing term. Thus, the number of features in these feature vectors is equal to the vocabulary size. Due to the inherent properties of textual data, there are usually thousands or even tens of thousands of unique terms in the vocabulary. As each unique term represents a new dimension in the feature space, the set of feature vectors is of very high dimensionality.

Because of the high dimensionality in the feature space, the feature vectors are not suitable as input to the text classifier since the scalability will be poor [1, 19]. In order to improve the scalability of the text categorization system, we propose that dimensionality reduction techniques should be employed to reduce the dimensionality of the feature vectors before they are fed as input to the text classifier. We have proposed and studied four dimensionality reduction techniques applicable to the text categorization problem. We will discuss these techniques in detail below.

5.4.1 The DF Method

As it has been pointed out in section 5.1.3, feature selection is one of the techniques for dimensionality reduction. By selecting a subset of the most important features from the initial feature set, the number of dimensions in the feature space can be greatly reduced.

The first technique for dimensionality reduction to be examined is called the DF method. The DF method is a feature selection technique specifically for the text categorization task. Given the set of training documents and their corresponding category labels, the DF method is applied to reduce the vocabulary size by term selection based on a local term ranking technique.

In the training document set, a set of training documents is provided together with a specification of which of the pre-defined categories each training document belongs to. In the DF method, the categorization information is used for grouping the training documents such that all the documents belonging to the same category are put into the same group. When there are overlaps between the categories, a document may belong to more than one group. After the documents are grouped, we can then form groups of the indexing terms in the vocabulary by putting in a group all terms contained in documents belonging to the same category. This process results in a set of sub-vocabularies corresponding to each category.

For the purpose of term selection, the terms must be ranked according to some importance measure. In the DF method, this importance measure is based on the *document frequency* of each term within a document group. For each document group, the document frequency of a term is defined as the number of documents within that particular group containing the term. By choosing the document frequency as the importance measure, we are assuming that the important terms are those that appear frequently within a group of documents belonging to the same category. This is because the set of terms which are good representatives of the category topics should be used by most documents belonging to that category.

Based on the DF importance measure, terms are ranked separately within each sub-vocabulary. For term selection, a parameter d is defined such that within each sub-vocabulary, only the most important d terms with the highest ranks are selected. The sets of selected terms from each sub-vocabulary are then merged together to form the reduced feature set.

By adjusting the selection parameter d, we can control the dimensionality of the reduced feature vectors. A smaller d will result in fewer terms to be selected, thus higher reduction in dimensionality.

5.4.2 The CF-DF Method

From the discussion of the DF method, we can observe that terms that appear in most documents within the whole training set will always have a high within-group document frequency. Even though these frequently occurring terms are of very low discrimination value, and thus not helpful in distinguishing between documents belonging to different categories, they are likely to be selected by the DF method. Because of this, it is necessary to extend the DF method such that the discrimination value of a term is taken into account in the term selection process. Here, we will examine such a method, called the CF-DF method.

In the CF-DF method, a quantity called *category frequency* (CF) is introduced. To determine the category frequency of a term, the training documents are grouped according to the categorization information, as in the DF method. For any document group, we say that a term appears in that group if at least one of the documents in that group contains that term. For any term in the vocabulary of the training document set, the category frequency is then equal to the number of groups that the term appears in.

By this definition, terms that are concentrated in a few categories will have a low category frequency, while those that are distributed across a large number of categories will have a high category frequency. The idea is that the discrimination value of a term can be measured as the inverse of its category frequency. In other words, we assume that terms that are good discriminators are most likely concentrated in a few categories, and should be considered more important as they are helpful in distinguishing between documents belonging to different categories.

In the CF-DF method, a two phase process is used for term selection. In the first selection phase, we define a threshold t on the category frequencies of the terms, such that a term is selected only if its category frequency is lower than the threshold t. In the second selection phase, the DF method is applied for further term selection to produce the reduced feature set.

5.4.3 The TFxIDF Method

In the DF method and the CF-DF method, the essential idea is to perform ranking of the terms in the vocabulary based on some importance measure, such that the most important terms can be selected. In both of these methods, the key to minimize information loss as a result of term selection is to define a good importance measure so as to avoid filtering out terms that are useful for the text categorization task.

As is well recognized in the information retrieval community, a good measurement of the importance of a term in a document set is the product of the *term frequency* and the *inverse document frequency* (see section 5.3.4 for the definitions of term frequency and inverse document frequency), which is widely used in term-weighting based text retrieval systems. In order to examine the effectiveness of this measure for term selection, we propose to use the $TF \times IDF$ value to measure the importance of a term for term selection. In other words, the terms are ranked according to their $TF \times IDF$ values, and a parameter d is set such that only the d terms with the highest $TF \times IDF$ values are selected to form the reduced feature set.

5.4.4 Principal Component Analysis

As discussed in chapter 4, principal component analysis is a statistical technique for dimensionality reduction which aims at minimizing the loss in variance in the original data. It can be viewed as a domain independent technique for feature extraction, which is applicable to a wide variety of data. This is in contrast with the other three dimensionality reduction techniques we have discussed, which are domain specific feature selection techniques based on feature importance measures defined specifically for textual data.

Because of its generality, principal component analysis has been successfully applied as a preprocessor to the classifier in different classification tasks for dimensionality reduction of various types of input data, including speech signals [30], acoustic emission signals [63], electrocardiographic signals [40], and also handwritten character images [21]. In order to examine its effectiveness in reducing the dimensionality of textual data, principal component analysis is used for dimensionality reduction in the proposed text categorization model.

In order to perform principal component analysis on the set of training documents, the set of feature vectors produced from the documents are used to create a term frequency matrix:

$$\begin{pmatrix} tf_{11} & tf_{12} & \cdots & tf_{1K} \\ tf_{21} & tf_{22} & \cdots & tf_{2K} \\ \vdots & \vdots & & \vdots \\ tf_{N1} & tf_{N2} & \cdots & tf_{NK} \end{pmatrix}$$

where N is the number of documents in the document set, K is the number of indexing terms in the vocabulary, and tf_{ij} is the term frequency (TF) of the j^{th} indexing term in the i^{th} document. Using this term frequency matrix as the data matrix ($\mathbf{D}$), the principal components are computed as described in section 4.3.

5.5 The Neural Network Based Text Classifier

By means of dimensionality reduction techniques, the set of documents to be categorized is transformed into a set of feature vectors in a relatively low dimensional feature space. This set of reduced feature vectors is then fed to the text classifier as input. In the proposed model, a three-layer feed-forward network is used as the text classifier. In this section, we will discuss the details of the neural network based text classifier.

5.5.1 Network Topology of the Text Classifier

Figure 8 shows the topology of the neural network based text classifier. As shown in the figure, the neural network is a three-layer fully connected feed-forward network which consists of an input layer, a hidden layer, and an output layer. All neurons in the neural network are non-linear units with the sigmoid function as the activation function. In the input layer, the number of input units

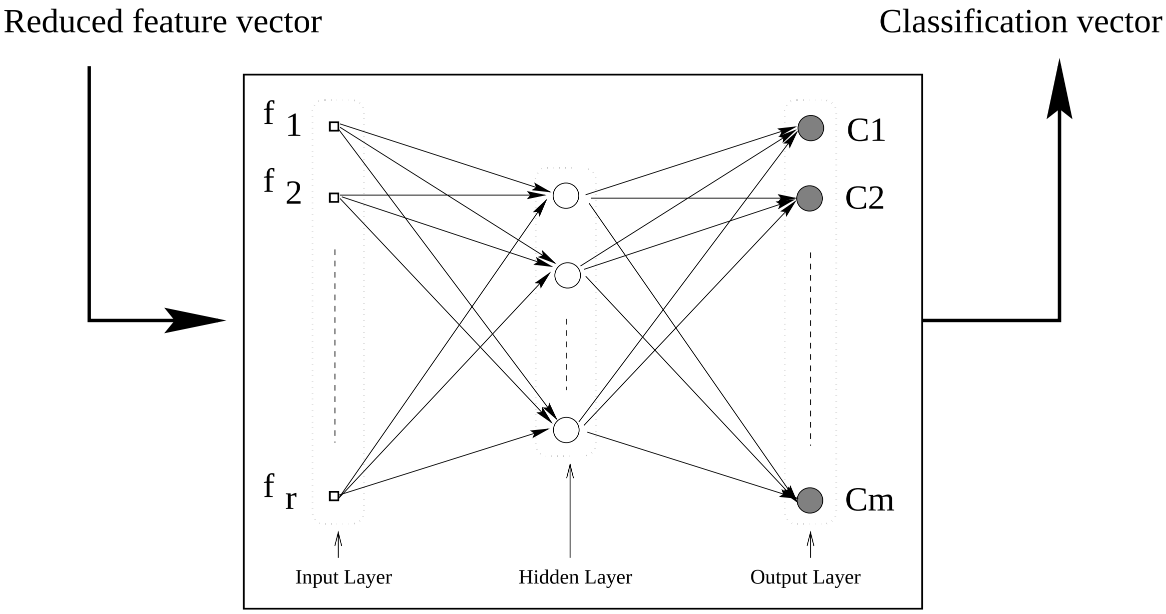

Figure 8: The neural network based text classifier.

(r) is equal to the dimensionality of the reduced feature space. In the output layer, the number of output units (m) is equal to the number of pre-defined categories in the particular text categorization task. The number of hidden units in the neural network affects the generalization performance [25, 48]. The choice depends on the size of the training set and the complexity of the classification task the network is trying to learn, and can be found empirically based on the categorization performance (see section 6.3.4).

For classification of the documents, reduced feature vectors representing the documents are fed to the input layer of the neural network classifier as input signals. These input signals are then propagated forward through the neural network so that the output of the neural network is computed in the output layer. As the sigmoid function is used as the activation function in the output units, the output of the neural network classifier is a real-valued classification vector with component values in the range [0, 1]. As explained in section 5.1.2, this real-valued classification vector represents a graded classification decision, in which the i^{th} vector component indicates the probability that the input document belongs to the i^{th} category. If binary classification is desired, a threshold can be set such that a document is considered to be belonging to the i^{th} category only if the i^{th} component of the classification vector is greater than the threshold.

5.5.2 Training the Text Classifier

The neural network classifier must be trained before it can be used for text categorization. Training of the neural network classifier is done by the Backpropagation learning rule based on supervised learning. In order to train the neural network, a set of training documents and a specification of the pre-defined categories the documents belong to are required. More precisely, each training example is an input-output pair:

$$T_i = (D_i, C_i)$$

where D_i is the reduced feature vector of the i^{th} training document, and C_i is the desired classification vector corresponding to D_i. The component values of C_i are determined based on the categorization information provided in the training set.

During training, the connection weights of the neural network are initialized to some random values. The training examples in the training set are then presented to the neural network classifier in random order, and the connection weights are adjusted according to the Backpropagation learning rule. This process is repeated until the learning error falls below a pre-defined tolerance level.

CHAPTER 6

PERFORMANCE EVALUATIONS AND EXPERIMENTAL RESULTS

In this chapter, we will explain the techniques for evaluating the performance of text categorization systems. Experiments were run using the Reuters-22173 test collection for text categorization to evaluate the performance of the text categorization model we proposed in chapter 5. We will present the experimental results in this chapter.

6.1 Performance Evaluation in Information Retrieval

Due to the large amount of research efforts spent on the text retrieval task, standard procedures are available for evaluating the performance of text retrieval systems, which are widely used by the information retrieval community. As many of the concepts in evaluating text categorization systems are adapted from the evaluation of text retrieval systems, an overview of these procedures will be given before we proceed to discuss the techniques for performance evaluation of text categorization systems.

6.1.1 Evaluating Text Retrieval Systems

As is the case with many other systems such as database management systems, the performance of text retrieval systems and information retrieval systems in general can be measured based on a number of different criteria, such as the execution time needed to perform a certain task, or the storage and memory overhead required by the system for execution. While these performance measures are common to many types of systems, most of the performance studies in information retrieval have been focused on measuring the retrieval effectiveness of text retrieval systems. In this section, we concern ourselves only with the evaluation of retrieval effectiveness for text retrieval systems.

As mentioned in chapter 1, the function of a text retrieval system is to select a subset of documents from a document database based on a user input query representing the information needs of a user. Thus, the retrieval effectiveness of a text retrieval system can be described based on the relevance of the retrieved documents as correspond to the user query. A text retrieval system is said to have high retrieval effectiveness if it is able to retrieve from the document database a set of documents that is considered highly relevant to the given query.

Unfortunately, the notion of relevance can be very subjective. Given a query and a corresponding document, different users may have different opinions on whether the document is relevant to the query. In order to solve this problem, the solution in most performance evaluation experiments for text retrieval systems is to base on a set of standard judgements for the decision of document relevance. This set of standard judgement is produced by a panel of domain experts, who will read a set of documents and judge the relevance of each document to each of the given queries [52].

In order to evaluate the retrieval effectiveness of a text retrieval system, the most common approach in information retrieval is to test the system using a test collection. A typical test collection includes three components: a set of test documents, a set of test queries, and a set of standard judgements for document relevance.

Figure 9 shows the typical experimental procedures for evaluating retrieval effectiveness using a test collection. As shown in the figure, given the set of test documents in the document database, the retrieval system under test will need to retrieve these documents according to the set of test queries. The retrieval results are then compared with the set of standard judgements to assess the ability of the system in retrieving documents that are relevant to the set of test queries.

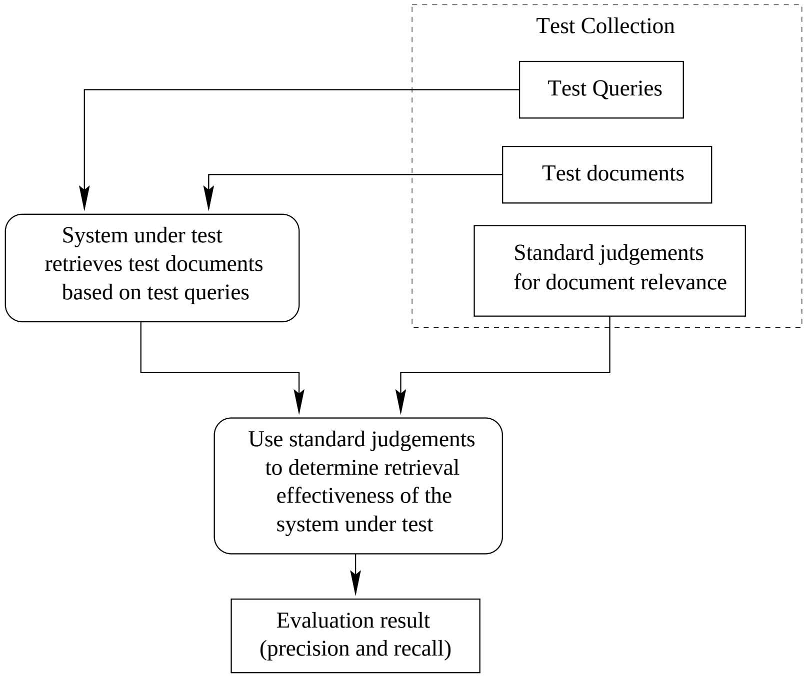

Figure 9: Evaluating text retrieval systems for retrieval effectiveness.

By this comparison, the retrieval effectiveness can be computed according to two commonly used measures, namely precision and recall:

Precision

Precision measures the accuracy of the retrieval result as indicated by the proportion of retrieved documents that are relevant. It is defined as:

$$Precision = \frac{\text{Number of retrieved documents that are relevant to the query}}{\text{Total number of documents retrieved}}$$

Recall

Recall measures how extensive the retrieval result is, as indicated by the proportion of relevant documents retrieved. It is defined as:

$$Recall = \frac{\text{Number of retrieved documents that are relevant to the query}}{\text{Total number of documents in the database that are relevant to the query}}$$

By definition, both precision and recall are limited to values in the range [0, 1].

For the set of retrieval results corresponding to the set of test queries, a set of precision and recall values can be computed. The overall performance can then be found by averaging, giving the average precision and recall for the system being tested.

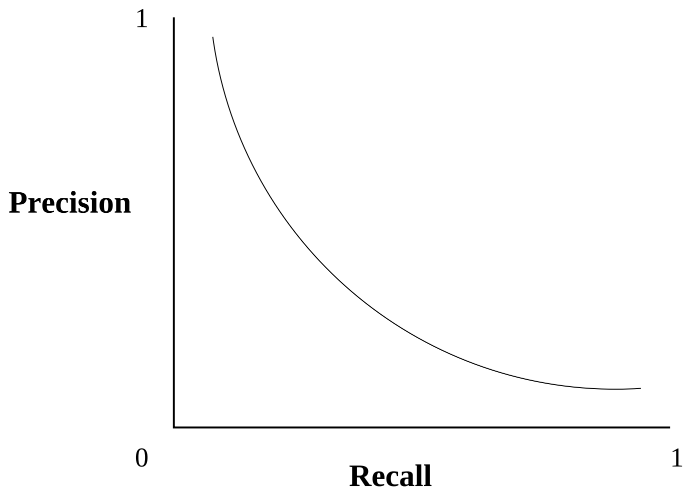

Figure 10: A typical precision-recall graph.

For text retrieval systems having a parameter which can affect precision and recall, different pairs of precision and recall values can be calculated by varying the parameter. A common approach is then to plot a precision-recall graph based on the set of pairs of precision and recall values. Generally, there is a tradeoff such that if recall is raised by varying the parameter, precision will drop, and vice-versa. Figure 10 shows the typical shape of a precision-recall graph for text retrieval systems. An illustration of how to produce the precision and recall graph for text retrieval systems that support document ranking can be found in [29].

Over the years, a number of test collections have been developed and used in the evaluation of text retrieval systems. Table 1 summarizes the statistics of some commonly used test collections. Since 1992, a very large test collection called the TIPSTER collection has also been developed as part of the Text REtrieval Conference (TREC) [16].

Collection	Subject	Number of test documents	Number of test queries
CACM	Computer Science	3204	64
CISI	Library Science	1460	112
CRAN	Aeronautics	1400	225
MED	Medicine	1033	30
TIME	General articles	425	83

Table 1: Test collections for text retrieval systems.

6.1.2 Evaluating Text Categorization Systems

Like text retrieval systems, performance of text categorization systems can be evaluated based on their categorization effectiveness. Just as retrieval effectiveness indicates the ability of a text retrieval system in retrieving relevant documents based on a user input query, categorization effectiveness indicates the ability of a text categorization system in providing accurate classification of text documents based on a set of pre-defined categories.

To evaluate the performance of text categorization systems, we need to define measures for the categorization effectiveness. One obvious approach is to redefine precision and recall in the context of text categorization. We can arrive at the definitions by looking at an analogy between the text retrieval task and the text categorization task. In text retrieval, the retrieval system has to make the decision of whether to retrieve a document or not, based on whether the document is relevant to a given query. This decision has to be made for each document and each different query. Similarly, a text categorization system has to make the decision of whether to assign a category to a document or not, based on whether the document is relevant to the topics represented by the category. This decision is made for each document and each pre-defined category. Based on this analogy, we can redefine precision and recall in the context of text categorization as follows:

$$Precision = \frac{\text{Number of category assignments that are made correctly}}{\text{Total number of category assignments made}}$$

$$Recall = \frac{\text{Number of category assignments that are made correctly}}{\text{Total number of category assignments that should be made}}$$

When there are more than one pre-defined category, there are two different ways for computing the precision and recall of the categorization system, referred to as *macroaveraging* and *microaveraging* in [31]. In macroaveraging, separate precision and recall values are calculated for each category, and then averaged over all categories to get the overall precision and recall. On the other hand, microaveraging calculates the precision and recall by considering assignment decisions for all categories at once.

Similar to the case of text retrieval, categorization effectiveness of text categorization systems can be evaluated by testing the system using a test collection designed for the categorization task. A test collection for text categorization systems should at minimal consists of these three components: a set of text documents, a set of pre-defined categories, and a specification of which of the pre-defined categories each document belongs to. The specification of category membership for each document is analogous to the standard judgements provided in test collections designed for the text retrieval task. To evaluate the categorization effectiveness, the categorization system under test will be used to categorize the set of documents based on the set of pre-defined categories provided in the test collection. Then the categorization result is compared with the specification given. By this comparison, the precision and recall can be computed.

In the next section, we will discuss one test collection designed specifically for the text categorization task, namely the Reuters-22173 collection. This collection has been used in our experiments to evaluate the categorization effectiveness of the proposed neural network based categorization model.

6.2 The Reuters-22173 Test Collection for Text Categorization

In this section, we will introduce the Reuters-22173 test collection, which is a test collection specifically designed for experiments with text categorization systems. Here, our intention is to give a brief overview of this test collection. Readers who have further interest in the test collection can find a thorough discussion of the preparation and various characteristics of the collection in chapter 8 of [32] [1]. We have conducted a number of experiments using the Reuters-22173 test collection for evaluation of the categorization model we proposed in chapter 5. We will present the results of these experiments in the next section.

6.2.1 Some History

The origin of the Reuters-22173 collection was a test collection used for the experiments done on a rule-based text categorization system called CONSTRUE, which was developed by Carnegie Group, Inc., for Reuters, Ltd. The collection was made available by Reuters, Ltd. and Carnegie Group, Inc., for research in the Information Retrieval Laboratory of the Computer Science Department at the University of Massachusetts at Amherst in 1990. Further formatting and data production was then worked on by various people [2]. The test collection is now freely available for research purposes [3].

6.2.2 The Test Documents

As its name suggests, the Reuters-22173 test collection contains a set of 22,173 documents. These documents are all full text articles distributed via the Reuters newswire during 1987, which include reports of recent events, long feature stories, quotes on market prices, and corporate earnings reports. The articles vary in length from a single line of text to multiple pages.

6.2.3 The Pre-defined Categories

There are a total of 689 categories defined in the Reuters-22173 collection. These categories are divided into 5 groups, namely TOPICS, ORGS, EXCHANGES,

[1] The information given for the Reuters-22173 collection in this section is based on the same chapter.

[2] See the README file that comes with the test collection for a list of people involved.

[3] The test collection can be downloaded by ftp at ciir-ftp.cs.umass.edu:/pub/reuters1.

PLACES, and PEOPLE [4]. The TOPICS group corresponds to categories with economic interest; the ORGS group corresponds to categories concerning important regulatory, financial, and political organizations; the EXCHANGES group contains categories about stock and commodity exchanges; the PLACES group contains categories representing different countries of the world; the PEOPLE group consists of categories about important political and economic leaders. These categories were assigned to the set of Reuters articles by two Reuters journalists stationed at Carnegie Group, Inc.

6.3 Experimental Results

In this section, we will present the results for the experiments we have conducted using the Reuters-22173 collection as the test collection for evaluating the performance of the categorization model we have proposed in chapter 5. Appendix B contains a full list of the experimental data obtained in the experiments.

We conducted three sets of experiments to explore different characteristics of the proposed model. The purpose of the first set of experiments was to compare the effectiveness of the 4 different dimensionality reduction techniques explained in section 5.4. In designing the second set of experiments, we intended to find out how different training sets affect the categorization effectiveness of the text categorization model. A third set of experiments was conducted with the purpose of finding the optimal size of the hidden layer for the neural network based text classifier.

Before we present the results, we will explain our experimental setup in detail.

6.3.1 Experimental Setup

In all of the experiments, we used a subset of the documents from the Reuter-22173 test collection for training and testing our text categorization model. We used a

[4] There is an additional group COMPANIES defined in distribution 1.0 of the test collection. This category group was not mentioned in [32] nor in the documentation that comes with the distribution.

set of 400 documents as the training set for training the text classifier by Back-propagation. For testing, a test set of 1,508 documents was used. This was with the exception of the experiments described in section 6.3.3, in which different training and test sets were used (see the section for more details). The set of test documents was kept unseen from the system during the training stage. In all experiments, we used a set of 10 categories from the TOPICS group defined in the Reuters test collection. Table 2 lists these 10 categories and the number of documents belonging to them.

Name of category	Number of documents
TOPICS:money-supply	192
TOPICS:nat-gas	133
TOPICS:soybean	124
TOPICS:livestock	118
TOPICS:copper	80
TOPICS:yen	79
TOPICS:jobs	78
TOPICS:rice	71
TOPICS:cotton	65
TOPICS:rubber	54

Table 2: The set of categories used in the experiments.

There are a total of 954 different documents belonging to one or more of the 10 chosen categories, with some overlaps between different categories so that some of the documents belong to more than one category. To select the set of documents for training, we randomly chose 200 documents from these 954 documents to be put into the training set. These 200 documents served as positive examples for training the neural network based text classifier. In order to introduce a set of negative examples into the training set, another 200 documents were randomly selected among the 21,219 documents not belonging to any of the 10 categories. These form a training set which consisted of 400 documents, with half of the documents being positive examples and the other half being negative examples. Table 3 shows the distribution of the training documents among the 10 categories.

In order to select documents for testing, we put into the test set the rest of the documents belonging to one or more of the 10 categories which were not selected

Name of category	Number of documents
TOPICS:money-supply	37
TOPICS:nat-gas	33
TOPICS:soybean	22
TOPICS:livestock	26
TOPICS:copper	15
TOPICS:yen	14
TOPICS:jobs	14
TOPICS:rice	18
TOPICS:cotton	17
TOPICS:rubber	12
Others	200

Table 3: Distribution of documents in the training set.

as positive examples in the training set. To test the ability of the categorization model in filtering out documents that should not have any category assigned, we randomly selected 754 documents that do not belong to any one of the 10 chosen categories and added them to the test set. This resulted in a test set of 1,508 test documents. Table 4 shows the distribution of the test documents among the 10 categories.

Name of category	Number of documents
TOPICS:money-supply	155
TOPICS:nat-gas	100
TOPICS:soybean	102
TOPICS:livestock	92
TOPICS:copper	65
TOPICS:yen	65
TOPICS:jobs	64
TOPICS:rice	53
TOPICS:cotton	48
TOPICS:rubber	42
Others	754

Table 4: Distribution of documents in the test set.

To create the set of indexing terms in the vocabulary, the 400 documents in the training set were processed by word extraction, stop words removal, and stemming. For word extraction, we defined a word as any consecutive character sequence contained in the character stream of a document, which starts with an alphabet ("a" – "z", "A" – "Z"), followed by any number of alphabets or digits. The end of a word is delimited by any non-alphanumeric character. Examples of acceptable words according to this definition include "john", "art", "db2", "b12", "a300s". The difference between upper case and lower case characters in the words was removed by converting all upper case characters to lower case (i.e. "A" $\rightarrow$ "a", "B" $\rightarrow$ "b", etc). After a set of words was extracted, stop words were removed based on a stoplist for general English text (the stoplist used can be found in appendix A). The remaining words were then stemmed using the Porter's stemming algorithm [43]. After stemming, we merged the sets of stems from each of the 400 training documents and removed the duplicates. This resulted in a set of 4,718 indexing terms in the vocabulary.

In order to create the set of initial feature vectors for representing the 400 training documents, we measured the term frequencies (TF) for each of the 4,718 indexing terms. The feature vectors were then formed by using the term frequency as the term weight. In other words, each of the feature vectors was a vector of the form:

$$D_j = < TF_{j,1}, TF_{j,2}, \ldots, TF_{j,4718} >$$

where $TF_{j,i}$ was the term frequency of the i^{th} indexing term in document j. This created a set of 400 feature vectors corresponding to each of the 400 training documents, where each feature vector was of dimensionality 4,718.

To reduce the dimensionality of these high dimensional feature vectors, different dimensionality reduction techniques were applied to reduce the dimensionality of the feature vectors from 4,718 to 328, with 93% reduction in dimensionality. This was done in all experiments except for the set of experiments explained in section 6.3.2 in which the reduced dimensionality was the varying parameter.

To create the set of classification vectors corresponding to each training document, the category membership information specified in the test collection was used. This created a set of 400 classification vectors corresponding to each of the 400 training documents. Each classification vector was a 10-dimensional vector of the form:

$$C_j = < c_{j,1}, c_{j,2}, \ldots, c_{j,10} >$$

where $c_{j,i}$ was set to 1 if document j belongs to the i^{th} category and was set to 0 if document j does not belong to the i^{th} category.

With the set of reduced feature vectors and their corresponding classification vectors, we formed input-output pairs of the form (D_j, C_j) so that each of these pairs represented a training example. We used this set of 400 training examples to train a three layer feed-forward neural network using Backpropagation. The neural network had 328 input units, corresponding to the reduced dimensionality (except for the experiments in section 6.3.2). The output layer had 10 units, in which the output of each unit corresponded to the assignment decision for each of the 10 categories. Except for the set of experiments explained in section 6.3.4 where the hidden layer size was the varying parameter, the number of hidden units was set to 25. During training, training examples were presented to the network in random order, until the learning error was less than 0.01. Note that up to this stage, no test documents from the test set were used in any way for training or dimensionality reduction.

To test the trained text classifier, we need to create a set of reduced feature vectors for the documents in the test set to be fed as input to the neural network classifier. To do this, we processed the 1,508 test documents in the same way as the training documents by word extraction, stop words removal and stemming. This created a set of terms from the test documents. In order to keep the vocabulary the same as that of the training documents, we dropped all terms from the test documents that were not in the vocabulary formed from the training documents. Term frequencies were then measured for each of the remaining terms. By this, we created a set of 1,508 initial feature vectors corresponding to the set of test documents.

We then need to reduce the dimensionality of these initial feature vectors created from the test documents. When the DF method, the CF-DF method and the TFxIDF method were used to reduce the dimensionality of the training feature vectors, dimensionality of the test feature vectors was reduced simply by dropping any dimension not corresponding to any of the dimensions in the reduced feature space of the training documents. In the case when principal component analysis was used for dimensionality reduction of the training feature vectors, we created the set of reduced feature vectors from the test documents by multiplying the high dimensional initial feature vectors with the set of eigenvectors found during the principal component analysis of the training documents. In other words, principal component analysis was not necessary on the set of test documents.

To test the performance of the trained text classifier, the set of reduced feature vectors created from the test documents was fed as input to it, and a set of output classification vectors was obtained. Each output vector was a 10-dimensional vector, with each component being a real number in the range [0, 1]. As we were interested in binary categorization only, we set a threshold at 0.5 such that if the i^{th} component of the output vector was greater than 0.5, we considered the decision of the text classifier was to assign the i^{th} category to the corresponding test document. This result was then compared with the category membership information specified for the test documents in the test collection, and the precision and recall were computed by macroaveraging.

All the experiments were conducted on Sun SPARCstation 20 machines running SunOS 4.1.4. Principal component analysis was performed using the matrix functions provided in MATLAB version 4.2c [35]. Neural network training and testing were done using the "trainbpx" function provided in the Neural Network Toolbox of MATLAB, with the momentum constant set to zero and the error goal (tolerance level) set to 0.01. For each run, the time needed for training and testing was within one hour.

This explains the experimental setup used in our experiments. We will now present the results obtained from the experiments.

6.3.2 Comparison of Dimensionality Reduction Techniques

To compare the effectiveness of the proposed dimensionality reduction techniques, we conducted two experiments. In the first experiment, the three feature selection techniques for dimensionality reduction, namely the DF method, the CF-DF method and the TFxIDF method were compared. In the second experiment, we looked at the ability of principal component analysis in reducing the high dimensional feature space into a much lower dimensional space without adversely affecting the categorization effectiveness. In both experiments, we tested the ability of the various techniques in retaining important information in the original feature space by decreasing the reduced dimensionality until there was a sharp drop in categorization effectiveness.

Experiment 1

Figures 11 and 12 show the result of the first experiment. In this experiment, the DF method, the CF-DF method, and the TFxIDF method were used to reduce the dimensionality of the initial feature vectors. The dimensionality of the reduced feature space was varied among 4,000, 3,000, 2,000, 1,000, 500 and 328.

As shown in the figures, the TFxIDF method was the best among the three techniques in retaining important terms from the original vocabulary. This was observed by the fact that the method was able to reduce the dimensionality of the original feature space to 1,000 before there was a sharp drop in precision and recall due to the loss of terms important for the categorization task. This represented a reduction rate of 78.8%. The CF-DF method was second in effectiveness among the three. It was able to reduce the dimensionality by 36.4% to 3,000 before the precision and recall dropped dramatically. The least effective among the three was the DF method, which was only able to reduce the dimensionality by 15.2% to 4,000. Further reduction resulted in a sharp drop in precision and recall. Figures 13 and 14 show replots of the precision and recall values against the dimensionality reduction rate.

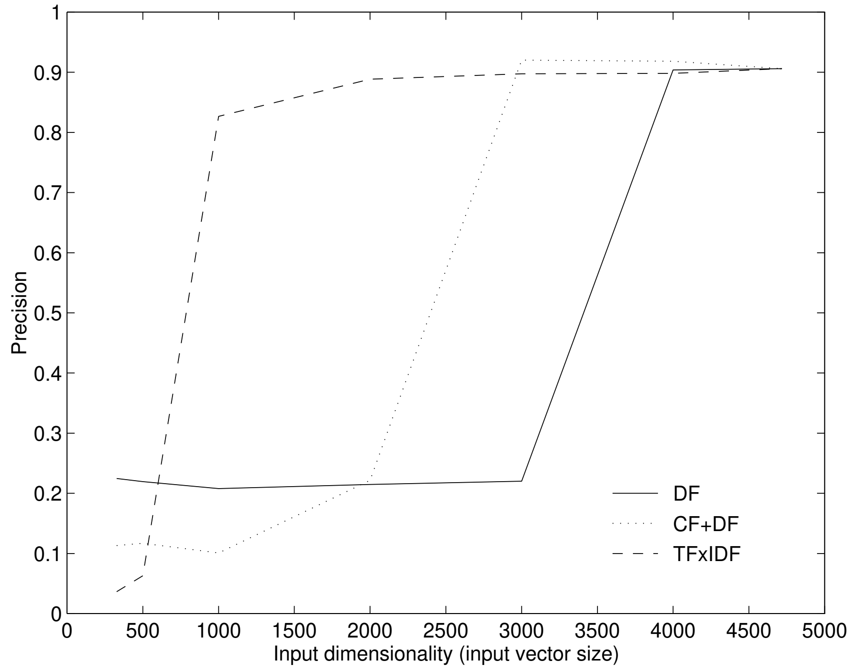

Figure 11: Precision and input dimensionality.

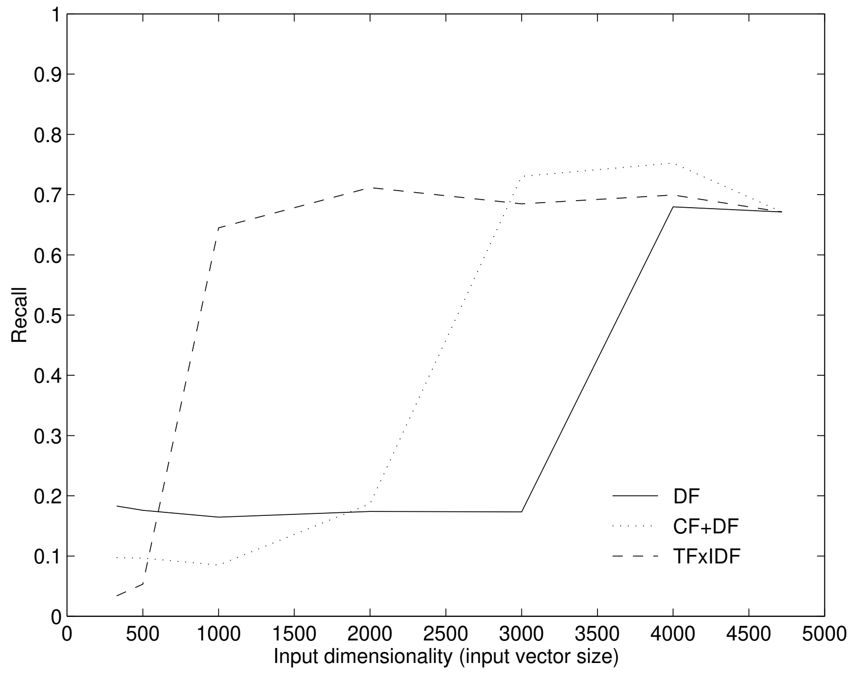

Figure 12: Recall and input dimensionality.

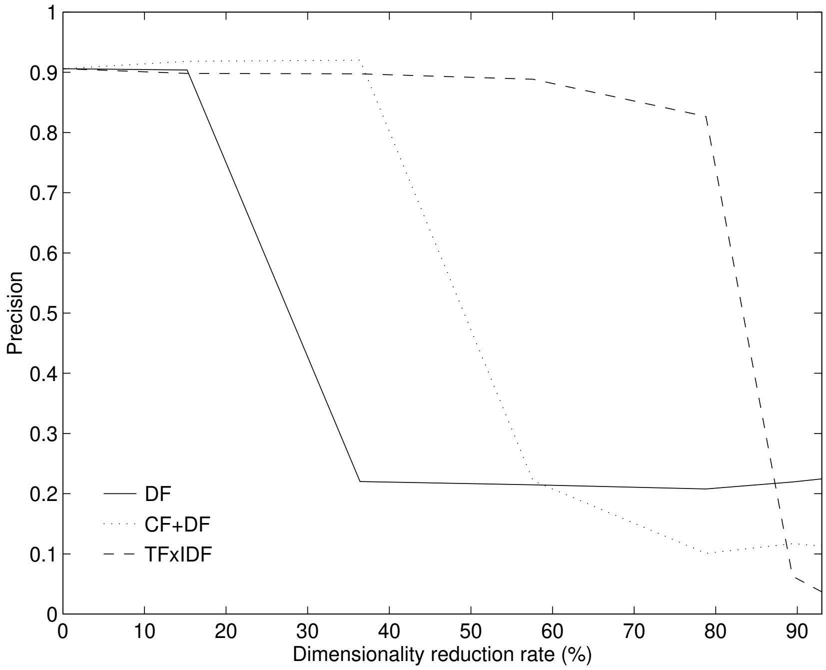

Figure 13: Precision and rate of reduction in dimensionality.

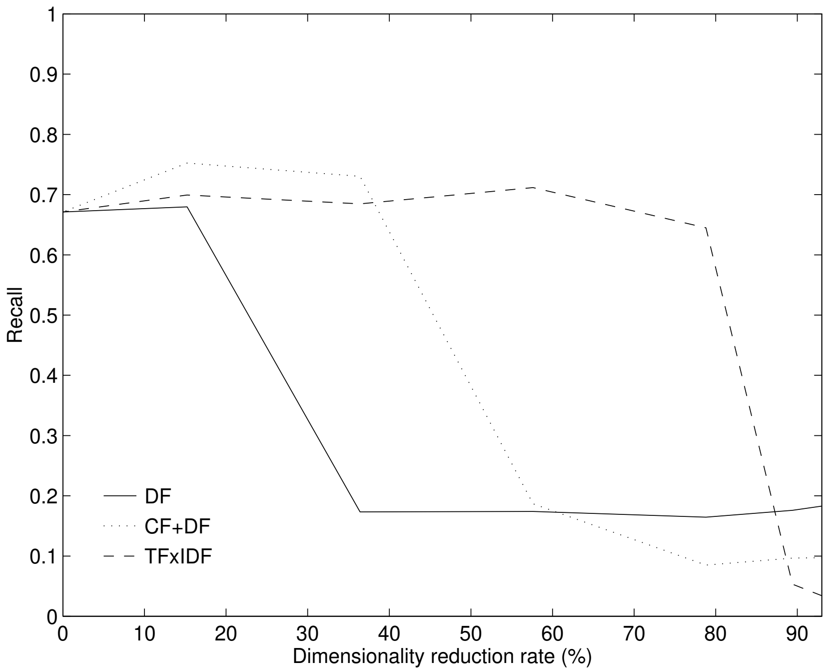

Figure 14: Recall and rate of reduction in dimensionality.

An implication of the results obtained in experiment 1 is that the $TF \times IDF$ score is a good measure for term importance, which was not surprising given its proven effectiveness for term weighting in text retrieval. Comparing the results obtained for the DF method and the CF-DF method, it was concluded that the addition of the category frequency (CF) was effective in helping to screen out terms of very low discrimination values. However, the effectiveness was not as significant as when the $TF \times IDF$ score was used.

Experiment 2

In the second experiment, principal component analysis was tested for its ability in encoding important information from the high dimensional feature space into a reduced feature space of much lower dimensionality. We used similar testing methodology as in experiment 1, and varied the reduced dimensionality among 350, 328, 200, 100, 50, 30, 10, and 5.

Figures 15 and 16 plot the precision and recall respectively against the reduced dimensionality. The results showed that principal component analysis was effective in reducing the dimensionality of the feature space from 4,718 to 50, and at the same time maintaining high precision and recall values. This represented a reduction rate of 98.9%, which was much higher than all of the three techniques we have tested in experiment 1. Replots of the precision and recall values against the reduction rate in dimensionality are shown in figures 17 and 18 respectively.

From the results of experiments 1 and 2, we concluded that principal component analysis was the most effective among the four proposed dimensionality reduction techniques in compressing the high dimensional term space into a much lower dimensional feature space, with minimal adverse effect in categorization effectiveness as measured by precision and recall. We also noticed that the neural network based text classifier was able to obtain reasonably high precision and recall values. This shows that neural networks trained by Backpropagation is effective in performing the text categorization task.

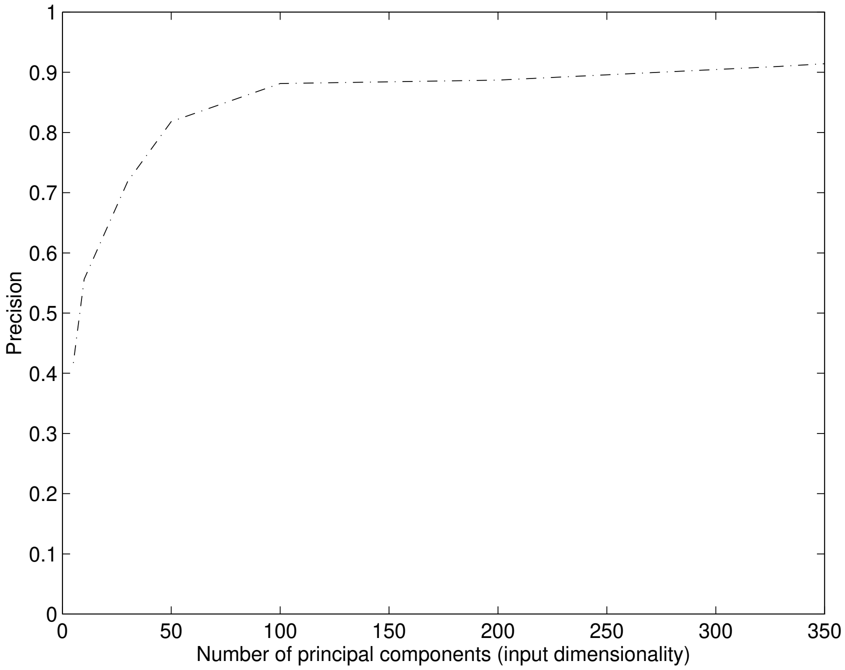

Figure 15: Precision and number of principal components.

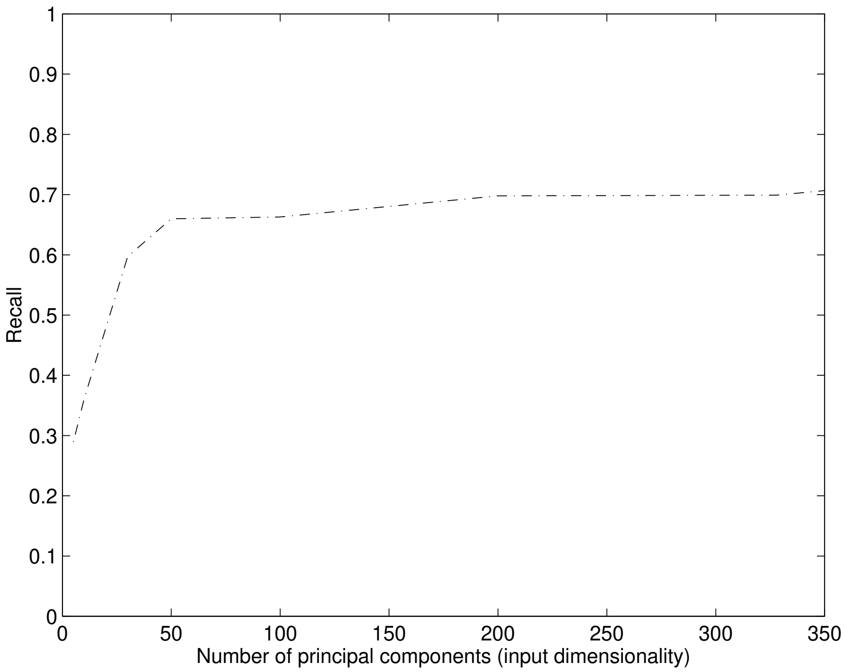

Figure 16: Recall and number of principal components.

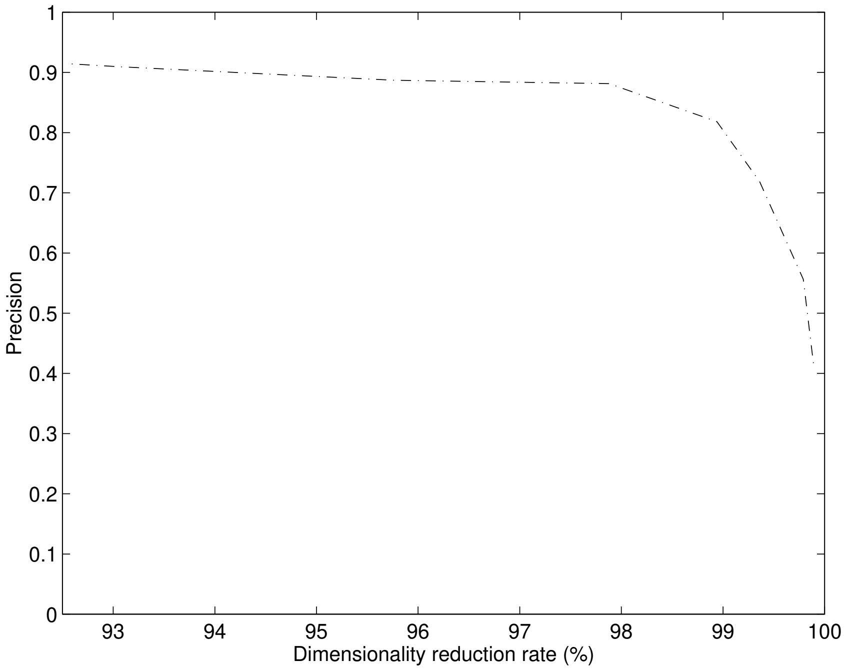

Figure 17: Precision and rate of reduction in dimensionality (PCA).

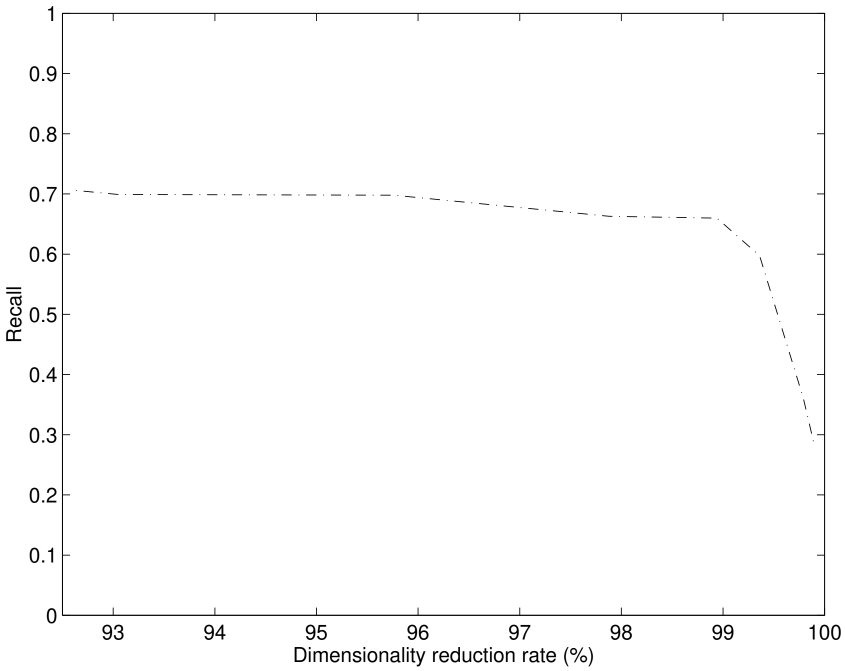

Figure 18: Recall and rate of reduction in dimensionality (PCA).

6.3.3 Varying the Training Set

In order to find out how varying the size and composition of the training set will affect categorization effectiveness, two experiments were conducted. In the first of these experiments (experiment 3), we varied the size of the training set by incrementally removing randomly selected training examples from the original training set of 400 examples. In the other experiment (experiment 4), the composition of the training set was varied by changing the ratio of positive and negative examples without changing the overall size of the training set.

Experiment 3

The purpose of conducting this experiment was to find out how the size of the training set affects precision and recall. In the experiment, the size of the training set was decreased stepwise from 400 to 50, with a decrement of 50 in each step. Starting with the original training set of 400 training examples, we randomly selected 50 training examples in each step and removed them from the current training set to form the next smaller training set. Precision and recall were computed for each training set in each step. This was done until a training set of only 50 training examples was formed in the last step.

Besides varying the size of the training set, we also compared the precision and recall for five different settings in which either no dimensionality reduction was done [5], or dimensionality reduction was done by the four proposed dimensionality reduction techniques to form a reduced dimensionality of 328.

Figures 19 and 20 show the results of the experiment. As shown in the figures, poor precision and recall were obtained when the DF method, the CF-DF method and the TFxIDF method were used for dimensionality reduction. This was mainly due to the fact that the reduced dimensionality was kept low at 328 dimensions. From the figures, we also observe that the precision and recall obtained when principal component analysis was used for dimensionality reduction were very

[5] The input to the neural network classifier in this case was the high dimensional feature vectors with dimensionality equals to 4,718.

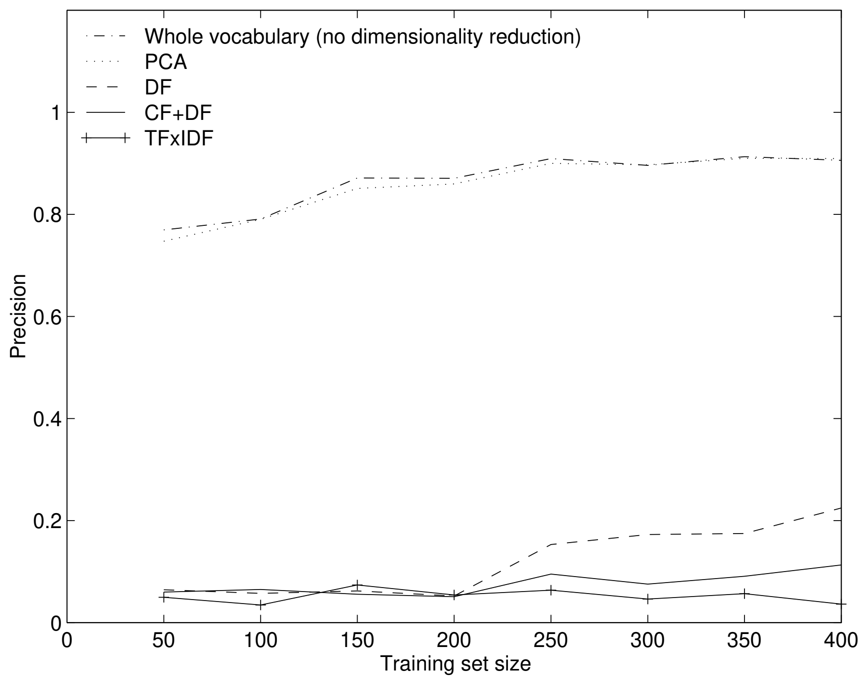

Figure 19: Precision and size of training set.

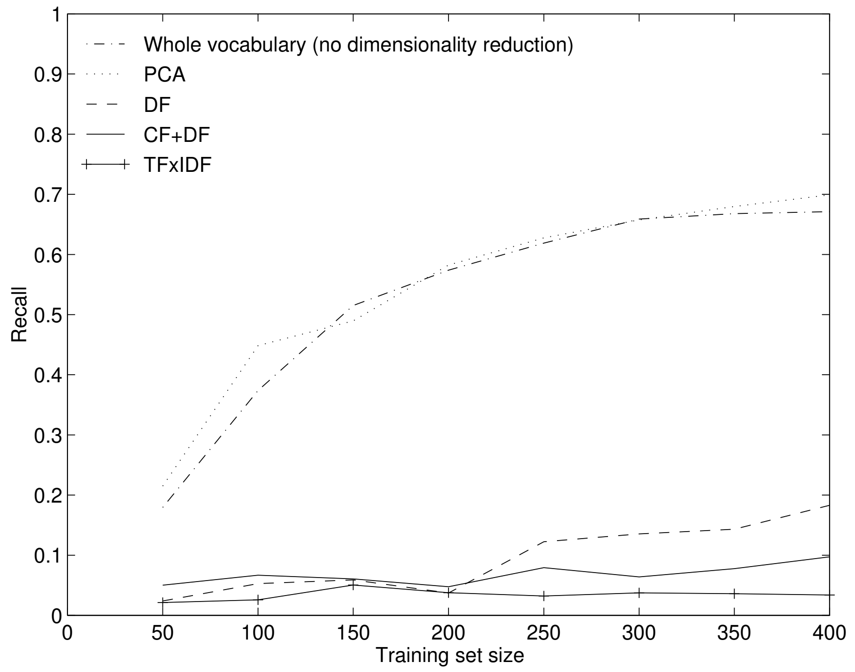

Figure 20: Recall and size of training set.

close to those obtained when there was no reduction in dimensionality. The implication is that principal component analysis was able to encode most of the information in the original feature space that are useful for text categorization in a much lower dimensional space.

When the size of the training set was decreased, both precision and recall were reduced. However, we observe that the reduction in precision was much smaller than the reduction in recall when the size of the training set was decreased. Recall was reduced as less number of training examples were presented to the neural network, many test documents belonging to the categories but were not similar to the training documents were missed. On the other hand, precision remained high as the neural network assigned categories to test documents similar to the training documents, and most of these assignments were correct. The much higher sensitivity of recall to variations in the size of the training set implies that a large training set should be used in order to ensure high recall values. On the other hand, in situations where precision is more important, a relatively small training set can be used.

Experiment 4

In this experiment, the composition of the training set was changed by varying the ratio of positive and negative examples in the training set. To do this, we started with a training set with all 400 training examples being positive examples. In other words, each training example belonged to at least one of the 10 chosen categories. From this training set, we decreased the number of positive training examples incrementally, from 400 to 200, with a decrement of 40 in each step. During each step, 40 positive examples were randomly chosen from the current training set and removed. These were replaced by 40 negative examples, which were documents not belonging to any of the 10 categories. In this way, we were able to change the ratio of positive and negative examples without changing the overall size of the training set. Precision and recall were computed in each step.

Due to the way we created the training set in this experiment, we were not

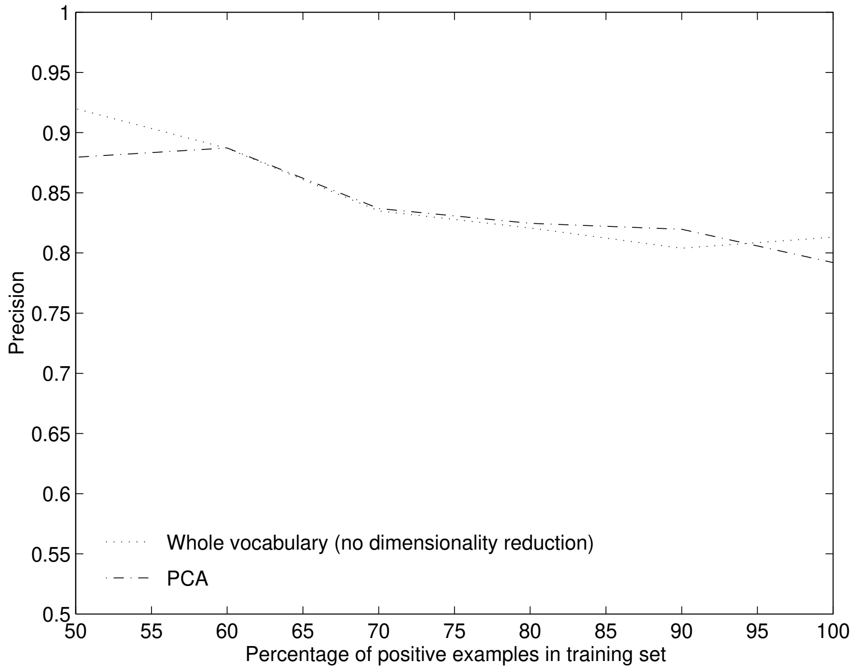

Figure 21: Precision and percentage of positive examples in training set.

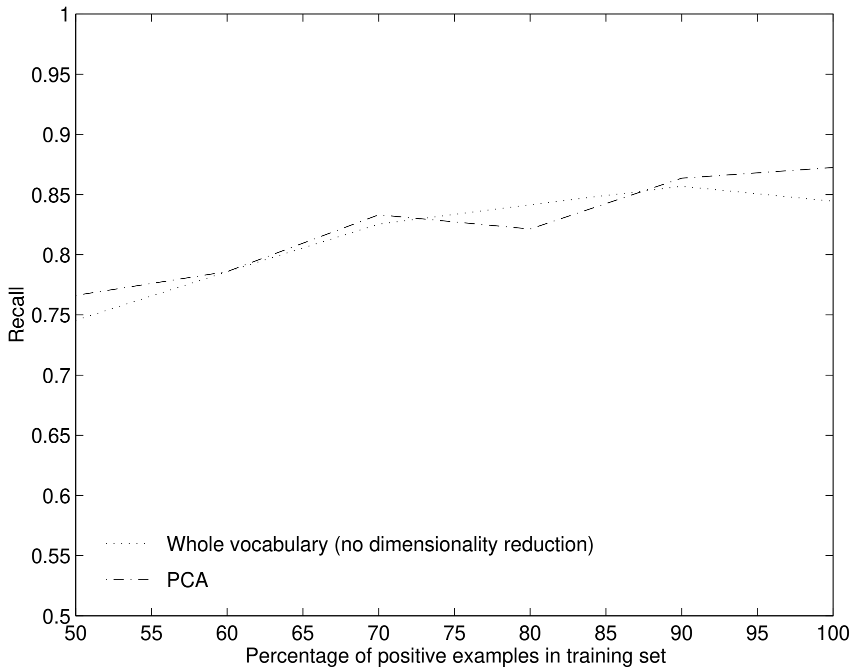

Figure 22: Recall and percentage of positive examples in training set.

able to use the original test set of 1,508 test documents that was used in other experiments. Instead, a test set of 1,108 documents was used in this experiment. Table 5 shows the distribution of documents in this test set.

Name of category	Number of documents
TOPICS:money-supply	110
TOPICS:nat-gas	75
TOPICS:soybean	81
TOPICS:livestock	63
TOPICS:copper	45
TOPICS:yen	51
TOPICS:jobs	43
TOPICS:rice	39
TOPICS:cotton	35
TOPICS:rubber	32
Others	554

Table 5: Distribution of documents in the test set used in experiment 4.

The results of the experiment are shown in figures 21 and 22. As shown in the figures, recall increased with an increase in the percentage of positive examples in the training set. However, the opposite was true for precision.

The results show that increasing the number of positive examples in the training set was important for improving recall. However, the presence of negative examples in the training set was also important for improving precision. This can be explained by the fact that negative training examples can improve the ability of the text categorization system in screening out documents not belonging to any of the pre-defined 10 categories. This represents a tradeoff between improving recall and improving precision when the ratio of positive and negative examples in the training set was changed.

6.3.4 Optimal Size of the Hidden Layer

As mentioned in section 5.5, the number of hidden units in the hidden layer affects generalization performance of the neural network. In general, it was found that simple neural networks generalize to unseen data better than more complex networks [60]. In order to maximize the generalization performance of a neural network, one possible strategy is therefore to minimize the network complexity by minimizing the size of the hidden layer.

Experiment 5

The purpose in conducting this experiment was to find the optimal size of the hidden layer for better generalization performance. Generalization performance in this case was indicated by the precision and recall obtained in categorizing the documents in the test set. In this experiment, the number of hidden units was varied among 2, 5, 7, 10, 15, 25, 35, 45, and 55. We computed the precision and recall for each of these settings, and the optimal size of the hidden units was chosen as the one which maximized precision and recall.

The results of the experiment are shown in figures 23 and 24. As can be seen from the figures, precision and recall dropped dramatically when the number of hidden units was less than 7. Recall was maximized when the size of the hidden layer was 15, while at the same time precision was saturated. We therefore concluded that the optimal size of the hidden layer should be 15.

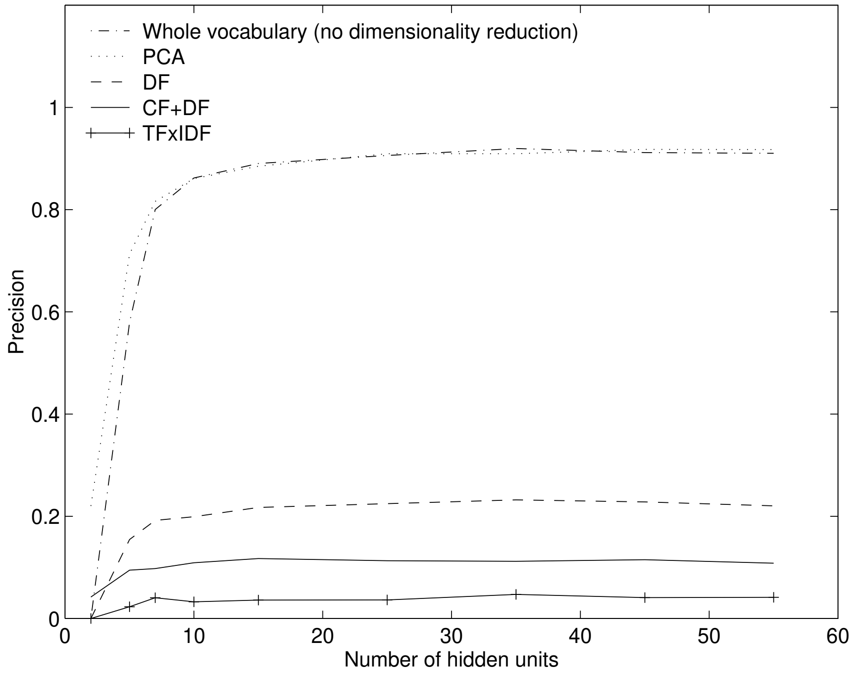

Figure 23: Precision and size of hidden layer.

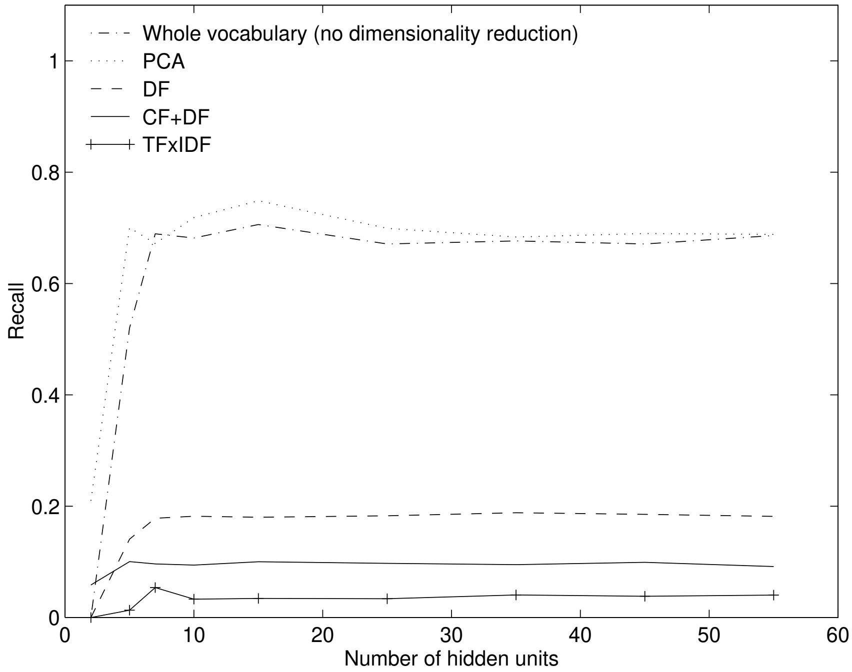

Figure 24: Recall and size of hidden layer.

CHAPTER 7

CONCLUSIONS

With the rapid growth in the amount of electronically stored textual data, information overloading is becoming a serious problem. The popularity of text intensive Internet services such as the World Wide Web, USENET news and electronic mails has resulted in a rapid increase in the number of online information retrieval systems trying to help the users find the relevant information they need. Without such systems, finding the information that one needs is often difficult and time consuming given the huge amount of information being put online. In order to solve the information overloading problem, it has been the aim of information retrieval researchers to derive ways to enable faster and easier access to information relevant to the needs of a user.

The information retrieval research community has previously concentrated on using statistical and probabilistic techniques for solving the information retrieval tasks, including text retrieval, text filtering and text categorization. Among these tasks, research efforts have been focused on text retrieval. These efforts have resulted in a number of mature models such as the vector space model for text retrieval [50]. These models are capable of solving real-world text retrieval problems with reasonably good performance, and are being applied in some World Wide Web text retrieval services such as Lycos, Alta Vista, and Infoseek. Despite this, retrieval performance is far from perfect. The situation is much worse for text categorization, which has received much less attention when compared to text retrieval. The lack of good automatic text categorization algorithms is evident in World Wide Web text categorization services, such as Yahoo, where the categorization task is mainly done manually.

In order to improve the performance of information retrieval systems, more and more researchers have recently turned to artificial intelligence techniques, among which artificial neural networks have been given more and more attention due to their increasing success in other domains.

Motivated by the above trend, this research has concentrated on the application of artificial neural network techniques to the text categorization task. In particular, we chose to focus on the Backpropagation neural networks because of their relative success and popularity in other domains. To better utilize the neural network model, we formulated text categorization into a text classification task. We used the neural network as the text classifier to classify a given set of text documents into a number of pre-defined classes or categories.

In the course of our research, we noticed that the major hindrance to the application of neural networks in the text categorization task was the high dimensionality of the feature space needed to represent textual data. This high dimensionality has been shown to cause scalability problems to neural networks [1, 19]. In order to solve this problem, we proposed and compared four different dimensionality reduction techniques to reduce the feature space into an input space of much lower dimension for the neural network.

7.1 Contributions

The contributions of this thesis are summarized as follows:

- A classification model based on the popular Backpropagation neural network was proposed for the text categorization task.
- Categorization effectiveness and feasibility of the proposed model was tested empirically. Experiments were conducted using the proposed model to categorize real-world full-text newswire articles contained in the Reuters-22173 test collection for text categorization. The results showed that Backpropagation learning in neural networks was able to give good categorization performance as measured by precision and recall.
- To solve the scalability problem inherent in most applications of neural network techniques to information retrieval, four dimensionality reduction

techniques were proposed to reduce the high-dimension feature space typical for textual data into a low-dimension input space for the neural network. These techniques included three domain dependent term selection techniques, namely the DF method, the CF-DF method and the TFxIDF method, and one domain independent statistical data analysis technique called principal component analysis.

- Experiments were conducted to test the effectiveness of the four different dimensionality reduction techniques proposed. We showed by the results that all of the four techniques were able to reduce the input dimensionality by a significant amount. By comparing the four techniques, principal component analysis was found to be the most effective as measured by the amount of reduction in dimensionality. In the experiments conducted, an extremely high reduction rate of 98.9% was achieved by principal component analysis with insignificant decrease in categorization effectiveness.

- Because of the diversified background of researchers involved in the application of neural networks to information retrieval, there were considerable differences between previously proposed models. With the aim of promoting more research efforts in neural network based information retrieval, a summary was provided for some of the most interesting projects done. An attempt was made to classify the different proposals into two main classes based on whether or not pre-programming was present. We identified and discussed the differences and relative merits of each class of approaches.

7.2 Future Work

There are a number of directions that this research can be pursued further. Here we summarize these future research directions:

- To further test the effectiveness of the proposed model and to increase the generality of the empirical study, more extensive experiments should be conducted by using larger training and test sets.

- The categorization model proposed is general enough to accommodate most neural network models with the supervised learning paradigm. Possible models include linear neural networks such as the Adaline and newer variants to the original gradient descend based Backpropagation learning such as the conjugate gradient [59] and Rprop [45] algorithms. Further comparison study can be conducted to investigate the performance of each of these models.

- In this thesis, we used the term occurrence frequencies (TF) as the document features. One disadvantage of this approach is that longer documents tend to have higher TF and are favoured in the categorization process. A possible solution to this problem is to normalize TF by the document length. Experiments should be carried out to find out whether categorization performance will be increased by this variation.

- As mentioned in chapter 3, Latent Semantic Indexing (LSI) can be used to reduce the dimensionality of textual feature space [53]. LSI is similar to principal component analysis (PCA) in that dimensionality reduction is performed by extracting new features that are linear combinations of the original features in the high dimensional feature space. One difference between LSI and PCA is that in LSI, it is not necessary for the initial set of feature vectors to have zero mean. Moreover, the mathematics involved for generating the set of extracted features were different between LSI and PCA. Experiments can be conducted to compare the performance of LSI and PCA on dimensionality reduction for text categorization.

- Instead of using a single neural network for all pre-defined categories, an alternative is to use a "one network per category" approach. In this case, n neural networks are used for n pre-defined categories, with the output layer of each network having only a single output unit indicating the relevance of the input documents to a particular category. Each neural network can be viewed as a document filter for the corresponding category. The advantage is that it is easy to add more categories by training additional neural networks.

- Because of the popularity and rapid grow in size of the World Wide Web (WWW), there is a considerable amount of web pages being manually categorized at WWW text categorization services such as Yahoo. Due to the highly diversified topics in most web pages, the WWW is a good testing ground for the feasibility and effectiveness of the proposed model in solving practical text categorization problems in the real-world. A prototype system can be developed for automatic categorization of web pages using the existing set of categorized web pages as the training set [1]. As most of the existing WWW categorization services organize the set of categories in a hierarchical manner, modifications to the text categorization model are necessary to incorporate this. One possible solution is a multi-classifier approach such that different neural classifiers are used for different levels in the hierarchy.

[1] Due to obvious copyright issues, this can only be done with permissions from the corresponding categorization service provider.

APPENDIX A

THE STOPLIST

a	areas	better	downed	fact	going
about	around	between	downing	facts	good
above	as	beyond	downs	far	goods
across	ask	big	during	felt	got
after	asked	both	e	few	great
afterwards	asking	but	each	find	greater
again	asks	by	early	finds	greatest
against	at	c	eg	first	group
all	away	came	either	for	grouped
almost	b	can	else	former	grouping
alone	back	cannot	elsewhere	formerly	groups
along	backed	case	end	four	h
already	backing	cases	ended	from	had
also	backs	certain	ending	full	has
although	be	certainly	ends	fully	have
always	became	clear	enough	further	having
among	because	clearly	etc	furthered	he
amongst	become	co	even	furthering	hence
an	becomes	come	evenly	furthers	her
and	been	could	ever	g	here
another	before	d	every	gave	hereafter
any	beforehand	did	everybody	general	hereby
anybody	began	differ	everyone	generally	herein
anyhow	behind	different	everything	get	hereupon
anyone	being	differently	everywhere	gets	hers
anything	beings	do	except	give	herself
anywhere	below	does	f	given	high
are	besides	done	face	gives	higher
area	best	down	faces	go	highest

Table 6: The stoplist used for stop words removal (part 1).

him	knows	men	none	ordering	problem
himself	l	might	noone	orders	problems
his	large	more	nor	other	put
how	largely	moreover	not	others	puts
however	last	most	nothing	otherwise	q
i	later	mostly	now	our	quite
ie	latest	mr	nowhere	ours	r
if	latter	mrs	number	ourselves	rather
important	latterly	much	numbered	out	really
in	least	must	numbering	over	right
indeed	less	my	numbers	own	room
interest	let	myself	o	p	rooms
interested	lets	n	of	part	s
interesting	like	namely	off	parted	said
interests	likely	necessary	often	parting	same
into	long	need	old	parts	saw
is	longer	needed	older	per	say
it	longest	needing	on	perhaps	says
its	m	needs	once	place	second
itself	made	neither	one	places	seconds
j	make	never	only	point	see
just	making	nevertheless	onto	pointed	seem
k	man	new	open	pointing	seemed
keep	many	newer	opened	points	seeming
keeps	may	newest	opening	possible	seems
kind	me	next	opens	present	sees
knew	meanwhile	no	or	presented	several
know	member	nobody	order	presenting	shall
known	members	non	ordered	presents	she

Table 7: The stoplist used for stop words removal (part 2).

should	that	to	wanting	whereto	would
show	the	today	wants	whereunto	x
showed	their	together	was	whereupon	y
showing	them	too	way	wherever	year
shows	themselves	took	ways	wherewith	years
side	then	toward	we	whether	yet
sides	thence	towards	well	which	you
since	there	turn	wells	whichever	young
small	thereafter	turned	went	whichsoever	younger
smaller	thereby	turning	were	while	youngest
smallest	therefore	turns	what	whilst	your
so	therein	two	whatever	whither	yours
some	thereupon	u	whatsoever	who	yourself
somebody	these	under	when	whoever	yourselves
somehow	they	until	whence	whole	z
someone	thing	up	whenever	whom	
something	things	upon	whensoever	whomever	
sometime	think	us	where	whomsoever	
sometimes	thinks	use	whereabout	whose	
somewhere	this	used	whereabouts	whosoever	
state	those	uses	whereafter	why	
states	though	usual	whereas	will	
still	thought	usually	whereat	with	
such	thoughts	v	whereby	within	
sure	three	very	wherefrom	without	
t	through	via	wherein	work	
take	throughout	w	whereinto	worked	
taken	thru	want	whereof	working	
than	thus	wanted	whereon	works	

Table 8: The stoplist used for stop words removal (part 3).

APPENDIX B

EXPERIMENTAL DATA

Input Dimensionality	Reduction Rate (%)	DF Method		CF-DF Method		TFxIDF Method	
		prec.	recall	prec.	recall	prec.	recall
328	93.05%	0.2246	0.1829	0.1131	0.0973	0.0364	0.0339
500	89.40%	0.2194	0.1760	0.1169	0.0966	0.0629	0.0532
1000	78.80%	0.2079	0.1646	0.1008	0.0850	0.8266	0.6449
2000	57.61%	0.2147	0.1739	0.2217	0.1869	0.8885	0.7117
3000	36.41%	0.2201	0.1732	0.9202	0.7304	0.8975	0.6847
4000	15.22%	0.9040	0.6797	0.9183	0.7523	0.8982	0.6994
4718	0%	0.9060	0.6712	0.9060	0.6712	0.9060	0.6712

Table 9: Experimental data for experiment 1.

Number of Principal Components	Reduction Rate (%)	PCA	
		prec.	recall
5	99.89%	0.4172	0.2893
10	99.79%	0.5563	0.3610
30	99.36%	0.7188	0.5972
50	98.94%	0.8183	0.6599
100	97.88%	0.8814	0.6629
200	95.76%	0.8870	0.6980
328	93.05%	0.9095	0.6991
350	92.58%	0.9142	0.7067

Table 10: Experimental data for experiment 2.

Size of Training set	Whole Vocabulary		PCA		DF Method		CF-DF Method		TFxIDF Method	
	prec.	recall	prec.	recall	prec.	recall	prec.	recall	prec.	recall
50	0.7696	0.1794	0.7475	0.2149	0.0645	0.0238	0.0599	0.0501	0.0497	0.0213
100	0.7908	0.3737	0.7902	0.4483	0.0575	0.0529	0.0650	0.0668	0.0348	0.0257
150	0.8712	0.5151	0.8509	0.4896	0.0622	0.0589	0.0557	0.0605	0.0739	0.0505
200	0.8707	0.5739	0.8598	0.5824	0.0522	0.0372	0.0508	0.0476	0.0542	0.0377
250	0.9094	0.6189	0.8999	0.6275	0.1531	0.1223	0.0953	0.0792	0.0636	0.0323
300	0.8959	0.6589	0.8973	0.6574	0.1728	0.1354	0.0757	0.0639	0.0463	0.0374
350	0.9132	0.6678	0.9100	0.6798	0.1748	0.1432	0.0908	0.0777	0.0569	0.0361
400	0.9060	0.6712	0.9095	0.6991	0.2246	0.1829	0.1131	0.0973	0.0364	0.0339

Table 11: Experimental data for experiment 3.

Percentage of Positive Examples (%)	Whole Vocabulary		PCA	
	prec.	recall	prec.	recall
50%	0.8795	0.7662	0.9199	0.7455
60%	0.8873	0.7860	0.8870	0.7858
70%	0.8370	0.8332	0.8350	0.8254
80%	0.8246	0.8213	0.8207	0.8416
90%	0.8197	0.8636	0.8040	0.8568
100%	0.7920	0.8725	0.8132	0.8444

Table 12: Experimental data for experiment 4.

No. of Hidden units	Whole Vocabulary		PCA		DF Method		CF-DF Method		TFxIDF Method	
	prec.	recall	prec.	recall	prec.	recall	prec.	recall	prec.	recall
2	0	0	0.2202	0.2099	0	0	0.0421	0.0586	0	0
5	0.5794	0.5203	0.7130	0.7003	0.1546	0.1409	0.0946	0.1004	0.0230	0.0132
7	0.8002	0.6893	0.8162	0.6720	0.1920	0.1785	0.0978	0.0963	0.0407	0.0539
10	0.8620	0.6819	0.8608	0.7186	0.1991	0.1820	0.1091	0.0942	0.0327	0.0332
15	0.8903	0.7059	0.8849	0.7486	0.2173	0.1802	0.1172	0.1003	0.0361	0.0345
25	0.9060	0.6712	0.9095	0.6991	0.2246	0.1829	0.1131	0.0973	0.0364	0.0339
35	0.9196	0.6765	0.9095	0.6837	0.2321	0.1884	0.1120	0.0950	0.0471	0.0406
45	0.9116	0.6711	0.9180	0.6898	0.2280	0.1853	0.1150	0.0992	0.0410	0.0383
55	0.9103	0.6862	0.9176	0.6882	0.2204	0.1818	0.1082	0.0916	0.0414	0.0403

Table 13: Experimental data for experiment 5.

REFERENCES

[1] E. B. Baum and H. David, "What size net gives valid generalization?," *Neural Computation*, vol. 1, no. 1, pp. 151–160, 1989.

[2] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artificial Intelligence*, vol. 40, no. 1–3, pp. 235–282, 1989.

[3] H. Chen *et al.*, "Generating, integrating, and activating thesauri for concept-based document retrieval," *IEEE Expert*, vol. 8, no. 2, pp. 25–34, 1993.

[4] H. Chen *et al.*, "Automatic concept classification of text from electronic meetings," *Communications of the ACM*, vol. 37, no. 10, pp. 56–73, 1994.

[5] H. Chen and J. Kim, "Gannet: a machine learning approach to document retrieval," *Journal of Management Information Systems*, vol. 11, no. 3, pp. 7–41, 1994.

[6] H. Chen and T. Ng, "An algorithmic approach to concept exploration in a large knowledge network (automatic thesaurus consultation): symbolic branch-and-bound search vs. connectionist hopfield net activation," *Journal of the American Society for Information Science*, vol. 46, no. 5, pp. 348–369, 1995.

[7] C. Claverdon, J. Mills, and M. Keen, "Aslib cranfield research project: factors determining the performance of indexing systems," *ASLIB*, 1966.

[8] F. Crestani, "Learning strategies for an adaptive information retrieval system using neural networks," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1, pp. 244–249, 1993.

[9] F. Crestani, "An adaptive information retrieval system based on neural networks," in *Proceedings of the International Workshop on Artificial Neural Networks*, pp. 732–737, 1993.

[10] F. Crestani, "Learning strategies for an adaptive information retrieval system using neural networks," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1, pp. 244–249, 1993.

[11] S. Deerwester *et al.*, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[12] W. B. Frakes and R. Baeza-Yates, *Information Retrieval Data Structures and Algorithms*. Englewood Cliffs, New Jersey: P T R Prentice Hall, 1992.

[13] R. Gnanadesikan, *Methods for Statistical Data Analysis of Multivariate Observations*. New York: Wiley, 1977.

[14] J. L. Goldberg, "Cdm: an approach to learning in text categorization," in *Proceedings of the 7th International Conference on Tools with Artificial Intelligence*, pp. 258–265, 1995.

[15] D. Harman, "How effective is suffixing?," *Journal of the American Society for Information Science*, vol. 42, no. 1, pp. 7–15, 1991.

[16] D. Harman, "Overview of the third text retrieval conference (trec-3)," in *Proceedings of the Third Text REtrieval Conference*, pp. 1–20, 1994.

[17] W. Hersh *et al.*, "Ohsumed: an interactive retrieval evaluation and new large test collection for research," in *SIGIR '94: Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 192–201, 1994.

[18] P. Hingston and R. Wilkinson, "Document retrieval using a neural network," in *Proceedings of the 4th Australian Joint Conference on Artificial Intelligence*, pp. 304–310, 1990.

[19] G. E. Hinton, "Connectionist learning procedures," *Artificial Intelligence*, vol. 40, no. 1–3, pp. 185–235, 1989.

[20] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, no. 79, pp. 2554–2558, 1982.

[21] S. D. Hyman *et al.*, "Classification of japanese kanji using principal component analysis as a preprocessor to an artificial neural network," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, pp. 233–238, 1991.

[22] J. E. Jackson, *A User's Guide to Principal Components.* New York: Wiley, 1991.

[23] I. T. Jolliffe, *Principal Component Analysis.* New York: Springer-Verlag, 1986.

[24] J. Kivinen and M. K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors." Technical Report UCSC-CRL-94-16, Basking Center for Computer Engineering and Information Sciences, University of California, Santa Cruz, 1994.

[25] K. Knight, "Connectionist ideas and algorithms," *Communications of the ACM*, vol. 33, no. 11, pp. 59–74, 1990.

[26] T. Kohonen, *Self-Organization and Associative Memory.* Berlin: Springer-Verlag, 1984.

[27] K. L. Kwok, "A neural network for probabilistic information retrieval," in *SIGIR '89: Proceedings of the 12th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 21–30, 1989.

[28] K. L. Kwok, "A network approach to probabilistic information retrieval," *ACM Transactions on Information Systems*, vol. 13, no. 3, pp. 324–353, 1995.

[29] D. L. Lee, H. Chuang, and K. Seamons, "Effectiveness of document ranking and relevance feedback techniques." To appear in IEEE Software.

[30] T. Leen, M. Rudnick, and D. Hammerstrom, "Hebbian feature discovery improves classifier efficiency," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, pp. 51–56, 1990.

[31] D. D. Lewis, "Evaluating text categorization," in *Proceedings of the Speech and Natural Language Workshop*, 1991.

[32] D. D. Lewis, *Representation and Learning in Information Retrieval.* PhD thesis, University of Massachusetts, 1992.

[33] D. D. Lewis *et al.*, "Training algorithms for linear text classifiers." To appear in SIGIR '96.

[34] J. B. Lovins, "Development of a stemming algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11, no. 1–2, pp. 22–31, 1968.

[35] MathWorks, Inc., *MATLAB : high-performance numeric computation and visualization.* Natick, Mass.: MathWorks Inc., 1992.

[36] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, no. 5, pp. 115–133, 1943.

[37] G. V. Meghabghab and D. B. Meghabghab, "Multiversion information retrieval: performance evaluation of neural networks vs. dempster-shafer model," in *Intelligent Systems*, vol. 1, pp. 537–545, 1995.

[38] M. Minsky and S. Papert, *Perceptrons.* Cambridge, MA: MIT Press, 1969.

[39] K. J. Mock and V. R. Vemuri, "Adaptive user models for intelligent information filtering," in *Intelligent Systems*, vol. 1, pp. 507–516, 1995.

[40] J. Nadal and M. de C. Bossan, "Classification of cardiac arrhythmias based on principal component analysis and feedforward neural networks," in *Computers in Cardiology*, pp. 341–344, 1993.

[41] C. D. Paice, "Another stemmer," *SIGIR Forum*, vol. 24, no. 3, pp. 56–61, 1990.

[42] J. Platt, "Learning by combining memorization and gradient descent," *Neural Processing*, 1992.

[43] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[44] J. Reggia *et al.*, "A formal model for diagnostic inference," *Information Sciences*, vol. 37, pp. 227–285, 1985.

[45] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the rprop algorithm," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1, pp. 586–591, 1993.

[46] F. Rosenblatt, *Principles of Neurodynamics.* New York: Spartan Books, 1959.

[47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel distributed processing : explorations in the microstructure of cognition*, ch. 8, Cambridge, MA: MIT Press, 1986.

[48] D. E. Rumelhart, B. Widrow, and M. A. Lehr, "The basic ideas in neural networks," *Communications of the ACM*, vol. 37, no. 3, pp. 87–92, 1994.

[49] G. Salton, *Automatic Information Organization and Retrieval.* New York: McGraw-Hill, 1968.

[50] G. Salton, *The SMART Retrieval System — Experiments in Automatic Document Processing.* Englewood Cliffs, N.J.: Prentice Hall, 1971.

[51] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, no. 5, pp. 513–523, 1988.

[52] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval.* New York: McGraw-Hill, 1983.

[53] H. Schütze, D. A. Hull, and J. O. Pedersen, "A comparison of classifiers and document representations for the routing problem," in *SIGIR '96: Proceedings of the 18th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 229–237, 1995.

[54] G. Shafer and R. Logan, "Implementing dempster's rule for hierarchical evidence," *Artificial Intelligence*, vol. 33, pp. 271–298, 1987.

[55] G. Shafer, P. P. Shenoy, and K. Mellouli, "Propagating belief functions in qualitative markov trees," *International Journal of Approximate Reasoning*, vol. 1, pp. 349–400, 1987.

[56] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[57] I. Syu and S. D. Lang, "A competition-based connectionist model for information retrieval," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 5, pp. 3301–3306, 1994.

[58] H. Turtle and W. B. Croft, "Evaluation of an inference network-based retrieval model," *ACM Transactions on Information Systems*, vol. 9, no. 3, pp. 187–222, 1991.

[59] P. P. van der Smagt, "Minimisation methods for training feedforward neural networks," *Neural Networks*, vol. 7, no. 1, pp. 1–11, 1994.

[60] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight-elimination applied to currency exchange rate prediction," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 3, pp. 2374–2379, 1991.

[61] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *1960 IRE WESCON Convention Record*, (New York), pp. 96–104, 1960.

[62] R. Wilkinson and P. Hingston, "Incorporating the vector space model in a neural network used for document retrieval," *Library Hi Tech*, vol. 10, no. 1–2, pp. 69–75, 1992.

[63] J. Yang and G. A. Dumont, "Classification of acoustic emission signals via hebbian feature extraction," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, pp. 113–118, 1991.