COMBINING MACHINE LEARNING AND HIERARCHICAL STRUCTURES FOR TEXT CATEGORIZATION

by

Miguel Enrique Ruiz Ruiz

An Abstract

Of a thesis submitted in partial fulfillment of the
requirements for the Interdisciplinary Studies-Ph.D.
degree in Information Retrieval and Intelligent Systems (sponsoring
department-Computer Science) in the Graduate College of
The University of Iowa

December 2001

Thesis supervisor: Associate Professor Padmini Srinivasan

ABSTRACT

Text categorization is the process of algorithmically analyzing an electronic document to assign a set of categories (or index terms) that succinctly describe the content of the document. This assignment can be used for classification, filtering, or information retrieval purposes. Machine learning methods such as decision trees, inductive learning, neural networks, support vector machines, linear classifiers, k-nearest neighbor, and Bayesian learning have been applied to solve this problem but most of these applications ignore the hierarchical structure of the underling classification vocabulary.

This dissertation focuses on the use of hierarchical classification structures, such as the UMLS Metathesaurus or the Yahoo! hierarchy of topics, to build and train machine learning algorithms for text categorization. For this purpose we use a variation of the Hierarchical Mixtures of Experts (HME) model adapted for text categorization. We evaluate the HME model using neural networks, and linear classifier as the nodes of the hierarchy. We explore in detail the use of different feature and training set selection methods. Experimental results are reported using a large collection of MEDLINE documents (OHSUMED collection) to assess the effectiveness of the HME model for in text categorization.

COMBINING MACHINE LEARNING AND HIERARCHICAL STRUCTURES FOR TEXT CATEGORIZATION

by

Miguel Enrique Ruiz Ruiz

A thesis submitted in partial fulfillment of the
requirements for the Interdisciplinary Studies-Ph.D.
degree in Information Retrieval and Intelligent Systems (sponsoring
department-Computer Science)
in the Graduate College of
The University of Iowa

December 2001

Thesis supervisor: Associate Professor Padmini Srinivasan

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Miguel Enrique Ruiz Ruiz

has been approved by the Examining Committee for the thesis requirement for the Interdisciplinary Studies-Ph.D. degree in Information Retrieval and Intelligent Systems (sponsoring department-Computer Science) at the December 2001 graduation.

Thesis committee: ______________________
Thesis supervisor

____
Member

____
Member

____
Member

____
Member

To my wife Kenya, and my daughters Mariana and Andreina

ACKNOWLEDGEMENTS

I would like to thank to all the people who made my stay in Iowa one of the most cherish experiences of my life. To my advisor Padmini Srinivasan whose advise and support have been invaluable. To the members of my PhD Committee David Eichmann, Ted Herman, Gregg Oden and Alberto Segre for their support and helpful comments on my research. To all the faculty from the Department of Computer Science and the School of Library and Information Science specially Sriram Pemaraju, Don Alton, and Steve Bruell who have always been so helpful and supportive. To the people of the OISS specially Stephen Arum and Cherryl Mason for their support and kindness to me and my family. To my friends Herbert Hoeger, Sclaudina Vargas, Patricio Jarpa, Oswaldo Cadenas, Liana Marmol, Darío Almarza, Ramón Torres-Isea and Ethel Bontrager for their solidarity and support even during the most difficult times. I also would like to thank to the people from TextWise Labs at Syracuse, specially Edmund Yu and Páraic Sheridan for their support and encouragement. Thanks also to Fabrizio Sebastiani, Isabel Moulinier and Douglas Oard for their comments of my research during these years.

Finally, I would like to thank to my parents, the Esteva family especially Emperatriz and Bartolomé for their love and support and to my wife Kenya and my daughters Mariana and Andreina to whom this dissertation is dedicated.

ABSTRACT

Text categorization is the process of algorithmically analyzing an electronic document to assign a set of categories (or index terms) that succinctly describe the content of the document. This assignment can be used for classification, filtering, or information retrieval purposes. Machine learning methods such as decision trees, inductive learning, neural networks, support vector machines, linear classifiers, k-nearest neighbor, and Bayesian learning have been applied to solve this problem but most of these applications ignore the hierarchical structure of the underling classification vocabulary.

This dissertation focuses on the use of hierarchical classification structures, such as the UMLS Metathesaurus or the Yahoo! hierarchy of topics, to build and train machine learning algorithms for text categorization. For this purpose we use a variation of the Hierarchical Mixtures of Experts (HME) model adapted for text categorization. We evaluate the HME model using neural networks, and linear classifier as the nodes of the hierarchy. We explore in detail the use of different feature and training set selection methods. Experimental results are reported using a large collection of MEDLINE documents (OHSUMED collection) to assess the effectiveness of the HME model for in text categorization.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Classification is a task performed quite naturally by human beings. Classification was not invented, but is rather an innate human capability. It is related to our memory which is so essential for our adaptive behavior. Memory is organized in ways that make information gained from past experiences available for present situations. Although we experience only individual events, we remember them and identify them as instances of classes or categories. In consequence, the essence of memory organization is classification. How difficult will our lives be if we did not have classification skills? How many varied human activities have classification at their foundation? Even simple acts such as recognizing an entity as a human being and another as a bumble bee requires the ability to classify based on our perceptions. As mentioned by Estes [30] classification is indeed basic to all our intellectual abilities.

Categorization and classification are two terms that are used interchangeably by many authors, but as Estes [30] notes there are subtle differences between them. Classification implies that the collection of objects is partitioned into groups, but categorization carries a further implication that knowledge of the category to which an object belongs reveals something about the object's properties.

Categorization, which is the main concept of interest of this research, has been a major area of study in psychology, linguistics, and cultural anthropology. It is also important in biology, as for example in the area of taxonomy. Categorization is the process of assigning objects (of whatever kind) to categories (which are collections of objects that are grouped together for some purpose) [68].

My goal in this dissertation is to contribute to research on the automatic categorization of documents to conceptual categories. In particular I explore automatic categorization when the possible categories exhibit an inner hierarchical structure.

This chapter presents a brief review of categorization in psychology, linguistics, and cultural anthropology. My goal is to give a general background for categorization pointing out the similarities and the differences of this concept in different areas of study, as well as the connections between them.

1.1 Categorization in Psychology

Categorization can be viewed as a process, and also as the result obtained by this process. In psychology, categorization is believed to be an important part of concept formation. Two approaches have been developed in this area of concept formation:

- Hypothesis testing:

  In hypothesis-testing, concept formation is treated as a form of problem solving. This approach assumes that the problem for the learner is to formulate a hypothesis about the critical features and test the hypothesis against observations of a sequence of exemplars until an adequate hypothesis is discovered.

Solutions of categorization problems by trial-and-error hypothesis testing are quite characteristic of the kind of experience that a person has when learning to use a personal computer, a fax machine, or any other complex device. In these experiences the user faces problems that he/she cannot solve by reference to the cryptic instructions which are usually written in a technical terminology or style that is not familiar to the user. Therefore, the user tries to learn the operation of the device by using it, formulating mental hypothesis of the role of the components (probably following the instructions), and learning from the successes or failures until he/she familiarizes with the correct sequence of steps to operate the new device.

- Category learning:

  According to Estes [30], the definition of what constitutes category learning depends on the nature of the learning situation. There are two important aspects of category learning: a) whether the categories to be learned are finite or infinite, and b) if the learning situation may be expressed as a distinction between taxonomic (binary) and statistical (probabilistic) classifications. In taxonomic classifications, category membership is defined by critical features or attribute values, and all objects having the same description belong to the same category. For example classification of compounds according to their chemical properties as acid, base, or salt[1] and animals by species. In probabilistic categorizations,

[1] In chemistry an acid is defined as a proton (Hydrogen ion, H+) donor, a base is a proton acceptor, and a salt is the product of the reaction between an acid and a base.

rigid distinctions are not available, but there are statistical relations between features and categories with the property that the degree of some features or combinations of features makes an object more likely to belong to one category or another. Examples include medical diagnosis categories representing different varieties of schizophrenia. Probabilistic categorizations are learnable to the degree that the probability distributions differ between the categories.

1.1.1 Hierarchies

Even informal observation of every day categorization reveals that many objects fit into a number of categories. A single object might be called wire-haired terrier, terrier, dog, mammal or animal. On other occasions it might be called pet, friend, guard dog, or brute. Part of the power of human thought and reasoning arises from the ability to view the same concept in different ways, thereby allowing us to access different kinds of knowledge about it [95].

The hierarchical organization has been suggested as a particularly important method for organizing concepts. In fact, when people are asked to categorize an object in a neutral setting, without further instructions, they are very likely to provide one of the hierarchically-organized categories, such as terrier or dog, rather than a relatively stand alone category such as furry thing or something to be rescued from a fire. Thus, taxonomic categories, which are generally hierarchies, may be particularly important ones for thought and communication. In addition to the importance of the hierarchical organization, psychologists have long noted that a particular level of specificity of categories seems to be important. For example, people will normally refer to a wire-haired terrier as a “dog” rather than calling it a “terrier” or an “animal”. There seems to be something about the specific category dog that makes it just right for identification.

1.1.2 Properties of a Hierarchical Structure

Hierarchical structures possess several properties that make them ideal for representing categories. In general a hierarchy may be viewed as an upside down tree as shown in Figure 1.1. Categories that are higher in the hierarchy dominate or are “superordinate” to the lower level categories; the lower level categories are “subordinate” to the higher level ones.

A hierarchy is a kind of network. That is, it has nodes (categories) connected by relations. However, a hierarchy is a special kind of network. The only relation allowed between category members is the “set inclusion” relation. For example, the set of animals include the set of fish which includes the set of trout which includes the set of rainbow trout. The set inclusion relation is also called the IS-A relation. The nature of the IS-A relation is also important in determining the properties of hierarchies. First, the IS-A relation is “asymmetric”: all dogs are animals, but not all animals are dogs. Secondly, the category relation is transitive: all pines are evergreen, and all evergreens are trees; therefore all pines are trees. The transitivity of category membership generates a similar transitivity property ascription. Every property true of the members of a category is also true of the category’s subordinates. For example,

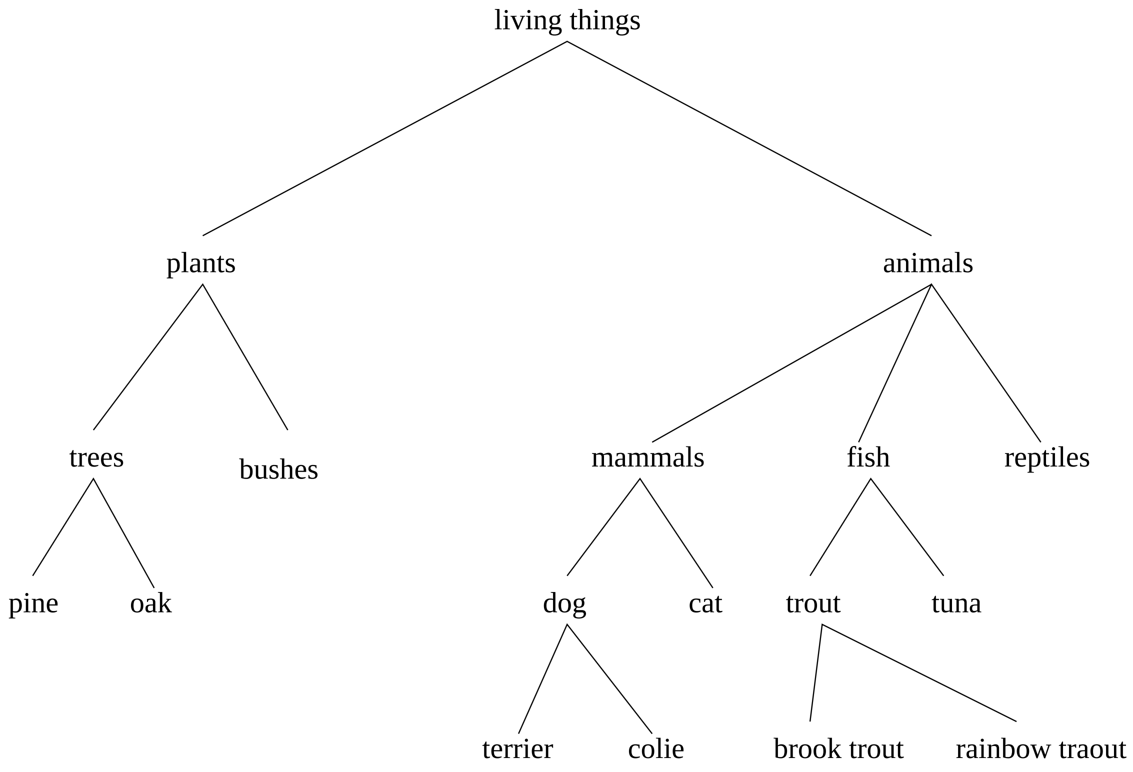

Figure 1.1: Example of a categorization hierarchy

suppose that all animals have blood, and therefore all dogs have blood, and therefore all terriers have blood.

1.1.3 Psychological Status of Hierarchies

Conceptual structures in humans are widely believed to have the general properties of hierarchies that we have just described. There is evidence that hierarchical structure appears to be a universal property of all cultures' categories of the natural world [3]. However, what is not clear is how hierarchies are mentally represented. There are two main possibilities that are not mutually exclusive. One possibility is that people's concepts are structured in memory like the diagrams in Figure 1.1. That is, perhaps concepts are connected in hierarchical networks, and the connections are used in order to make inductive inference and categorization judgments as described in the previous section. The second possibility is that the hierarchy results from a kind of reasoning process rather that being explicitly stored in memory. Suppose that you know that all Xs are Ys, and that all Ys are Zs. Now if you learn that all Zs have six fingers, what does it tells about Xs? A little thought will reveal that all Xs must also have six fingers, since all of them are Zs. Thus, even though the hierarchy may not have been stored in memory (before reading the paragraph), one could use the information about category inclusion to come to the correct answer. This suggests that people may not have a hierarchy stored in memory but may be able to infer category inclusion and then draw the appropriate inference. In other words, the category hierarchy could be pre-stored or it could be computed. If it is pre-stored, then the links in memory correspond to the IS-A links in Figure 1.1. If it is computed, then hierarchical relations are not stored in memory but calculated based on the properties of each pair of categories.

In the 1970's, many experiments were conducted to discover which of these accounts of conceptual structure was most accurate. Unfortunately, these experiments were not entirely conclusive. The main problem faced in any experiment that aims to prove either hypothesis is the set of assumptions about the memory structures and processes that are used in any particular task [95]. Since neither theory completely accounts for all of the observed data, each was modified in order to be more complete. The result was that it became difficult to tell the two views apart.

1.2 Categorization in Linguistics

Categorization in linguistics is considered an important process related to meaning. Linguists recognize that any cognitive activity includes categorization and conceptualization. To distinguish an object or event as such we need to identify it as similar or different from another object. Only then can we proceed to name it. Cognitive development of a child always involves an attempt to master the idea of same/different via some comparison of words and objects. For example, a child will ask questions such as: "a purse is a sort of a little bag, isn't it?"; "My teddy-bear - is he a bear?". These questions allow us to see the attempts of a child to structure his/her environment by finding for each object and its name an appropriate place in some naive taxonomy [35]. The same strategy is also commonly used in dictionaries to give definitions of words, *i.e.*, “machete: a heavy knife or cleaver used to cut down sugar canes, and as a weapon”.

When we conceptualize our view of the real world, we reveal various types of relations between objects. Many of the most important ones seem to be cultural universals. We cannot discuss any categorization problem without resorting to language. When we speak of the ontology of categorization, that is discussing the actual content of our mental activities, we cannot proceed without linguistic examples. Our ability to make generalizations is reflected in all languages, for example the opposition “general-specific” is present in all languages, thought their linguistic form varies from one language to another.

In the 1970’s the problem of general specific as well as its close relative “supercategory-category” was extensively researched in humanities. Probably the most relevant works to our review of categorization in linguistic are the experiments presented by E. Rosch and her collaborators [104]. Rosch’s approach to categorization in linguistic shows three basic hypotheses:

- Our environment is not chaotic but well structured. It is based on similarities and differences which belong to its ontology and not to our ability or inability to perceive and conceptualize. Hence categorization is conceived as mental activities that we develop to account for this ontology. Categorization helps us to assimilate this variety by conceptualizing some items as belonging to the same category, and others as those which belong elsewhere. According to Rosch there are two types of categories: natural categories, and semantic categories. Natural categories are those that depend on our perception, e.g., size, color, or shape. Semantic categories are conceptual, they are related to items that could be conceived as belonging to the same/different category on a non perceptual basics.

- Each category has an internal structure. According to Rosch, some members of a category may be better examples of that category than others. Rosch considers that the mentioned differences among members are ontologically based (based upon being or existence of the objects). That means that psychologically there are some members within each category that are more naturally identified. Those members that are more typically are called "salient" by Rosch.

- Any category has a member called the "prototype". This hypothesis is derived from the second hypothesis. The concept of a prototype presumes that the prototype includes the most typical features of a category, which enable us to identify the whole category starting from the prototype.

Rosch's experiments aimed to give support to these three basic hypotheses. Although the detailed discussion of her results is out of the scope of this work, we note the parallels between these hypotheses and the principles underlying feature selection and subset training set selection in machine learning algorithms.

1.3 Categorization in Cultural Anthropology

Cultural anthropologists have done extensive work studying categorization (or classification) in different cultures. Particularly works by Berlin and his collaborators [3] try to find the general principles of classification and nomenclature in folk biology. Berlin's major claim is that the observed structural and substantive topological regularities found among systems of ethnobiological classification of traditional people from many different parts of the world can be explained in terms of the human beings' similar perceptual and largely unconscious appreciation of the natural affinities among grouping of plants and animals in their environment. According to Berlin this groupings are recognized and named independently of their actual or potential usefulness or symbolic significance to humans.

There are two major approaches in ethnobiology the relativist approach and the comparativist approach. The relativist view adopts the position that cultures are different in many ways and that description of them provides only an imperfect and biased collection of facts that is highly influenced by the cultural preconceptions of the observer [29]. The comparativist approach, while recognizing the broad range of inter- and intra-cultural variation in human societies, seeks to discover and document general features of cross-cultural similarities that are widely shared and to develop theoretical explanations that underline the empirical generalizations one observes. Berlin's work [3] represents the comparativist approach while Ellen's work [29] represents the relativist approach.

The most significant account that Berlin reports in his work, is that hierarchical classifications are common in many cultures. Berlin reports the existence of the concept of higher-order and intermediate categories in cultures such as Maya, Kalam (from new Guinea), Tzeltal (from Mexico), Wayampi (from Brazil), Aguaruna (from the Peruvian Amazon), Ndumba (from New Guinea). Showing that even though all those cultures have been separated in distance and time they all share similar concepts and strategies in classifying animals and plants from their own regions. The reader is referred to Berlin [3] chapter 4 and Ellen [29] for a more detailed account of their approach.

The intent behind this brief overview is only to illustrate the attention given to the phenomenon of categorization by researchers representing different disciplines and viewpoints. In the following chapters I explore categorization in the context of machine learning algorithms developed to automatically categorize objects (documents) using a hierarchical collection of categories.

CHAPTER 2
BACKGROUND

2.1 Text Categorization

Text categorization, also known as *text classification* and occasionally as *topic spotting*, is the process of algorithmically analyzing an electronic document to assign a set of categories (or index terms) from a predefined vocabulary to succinctly describe the content of the document. This assignment can be used for classification, filtering, and retrieval purposes. Manual classification commonly referred to as indexing has been applied since the invention of writing to facilitate access to information. For years librarians have indexed documents using controlled vocabularies such as the Library of Congress Subject Headings, and the Medical Subject Headings (MeSH). The increasing amount of information available in different areas of knowledge creates the need to automate this process.

The origins of text categorization date to the 1960's when Luhn [80] proposed the use of statistical patterns of word occurrences in the title or abstract of documents to select predefined categories for indexing documents. The early literature in information retrieval also reveals the use of the term "automatic document classification". In general that was used to name three different tasks: (i) the automatic assignment of documents to a predefined set of categories [81], (ii) the automatic definition of document categories (currently known as clustering) [24], and (iii) the automatic assignment of uncontrolled vocabulary to documents, i.e., vocabulary extracted from the free text of documents (currently known as indexing) [6, 31]. During the 1980's most research efforts in text categorization concentrated on building categorization systems based on manually crafted decision trees and expert systems [4, 38, 43, 51]. Since the early 1990's researchers have explored a variety of machine learning methods for automatic text categorization [2, 5, 16, 26, 54, 63, 65, 66, 72, 78, 82, 84, 94, 96, 130, 141] generating most of the current literature in the field.

This chapter reviews the work in automatic text categorization and introduces the terminology, collections and performance measures used in this area.

2.2 Machine Learning and Text Categorization

Since computers were invented, we have wondered whether it is possible for them to learn how to perform specific tasks. Machine learning is an area of artificial intelligence that has been dedicated to this goal. Although it is not yet known how to program computers so that they can learn as well as people do, algorithms have been invented for certain types of learning tasks. Machine learning algorithms have proven to be very successful in solving many problems, for example, the best results in speech recognition have been obtained with such algorithms. Machine learning algorithms learn by performing a search on the solution space of the problem to be solved. For example, these algorithms use the *means/ends* principle in which at each step of the process the algorithm evaluates the different options (reachable states) and selects the one that moves it closer to the desired goal. Machine learning algorithms can be divided into two types *supervised* and *unsupervised* learning algorithms. Supervised learning algorithms operate by learning the objective function from a set of training examples and then applying the learned function to the target set. Unsupervised learning operates by trying to find useful relations between the elements of the target set. Our goal is to contribute to research on text categorization using supervised learning algorithms. In the next chapter alternative supervised learning methods are presented. In this chapter we consider this family of text categorization solutions more generally.

Text categorization can be characterized as a supervised learning problem. We have a set of examples (documents) that have been correctly categorized (usually by human indexers). This set is then used to train a classifier based on a machine learning algorithm. The trained classifier is then used to categorize the target set.

More formally, let $C = \{c_1, \ldots, c_n\}$ be a set of categories and $D = \{\mathbf{d}_1, \ldots, \mathbf{d}_N\}$ be a set of documents. Given a set of examples of the form $\langle \mathbf{d}_i, y_j \rangle$, where $\mathbf{d}_i \in D$, and if $\mathbf{d}_i \in c_j$ then $y_j = 1$, otherwise $y_j = 0$, the objective is to learn a function f such that $f(x) = 1$ if $x \in c_j$ and $f(x) = 0$ if $x \notin c_j$. This function is called the *classifier*.

This definition views a classifier as a single-label assignment function. Multi-label categorization with n categories $\{c_1, \ldots, c_n\}$ can be transformed into n independent problems of single-label categorization with categories $\{c_i, \overline{c_i}\}$, for each $i = 1, \ldots, n$. This assumes that the categories are statistically independent, i.e.

$f(d_j, c_k)$ does not depend on $f(d_j, c_{k'})$, for $k = 1, \ldots, n$; $k' = 1, \ldots, n$, and $k \neq k'$.

2.3 Document Representation

In order to generate a classifier for text categorization using a training set one must first derive a representation for each document. We represent documents using the vector space model as described by Salton [115]. First, all the words in the document are tokenized, filtered using a stop list (to remove common words such as articles, prepositions, common verbs, etc.), and stemmed. We use Lovins' stemmer for this [79]. Unique stems with their corresponding frequencies are kept. Each document is then represented by a vector:

$$\mathbf{d} = (x_1, \ldots x_m)$$

$$where \quad x_i = tf_i \times idf_i$$

$$and \quad idf_i = log\tfrac{N}{r_i}$$

Here m is the total number of unique stems (terms) in the training collection, x_i is the weight of the ith term in $\mathbf{d}$, tf_i is the ith term's frequency within $\mathbf{d}$, idf_i is its inverse document frequency[1], N is the total number of documents in the training set, and r_i is the number of documents in the training set that contain the term i. The vector space model allows us to represent a document as a real-valued vector that can be presented as an input to the machine learning algorithms. It should be noted that tf_i represents the importance of the term in the document while idf_i represents the importance of the term in the collection. Weighting strategies and their impact

[1]Note that there are several variations of idf. The formula presented here is used by the SMART retrieval system which is the tool we selected for preprocessing the documents.

on retrieval have been reviewed in [113].

2.4 Standard Test Collections

Experiments have been performed using several text collections of which three are now regarded as standard data sets for text categorization research: Reuters-1987 collection, OHSUMED collection, and AP TREC collection.

2.4.1 Reuters-1987 Collection

The Reuters-1987 collection[2] is a set of more that $20,000$ news wires stories from 1987. It is the most commonly used collection in text categorization research [2, 16, 43, 63, 65, 72, 96, 130, 136]. However, the different studies have not used the same criteria to split the collection into training and testing subsets, or the same criteria to treat the unlabeled data. They also differ in the number of categories used for evaluation. Four different versions of the Reuters-1987 collection have been used and the differences are shown in Table 2.1. The original Reuters-1987 collection (version 1 in Table 2.1) consisting of $22,173$ documents was provided by the Carnegie Group, Inc (CGI) who used it to evaluate the CONSTRUE system [43]. These documents are manually tagged with categories by personnel from Reuters Ltd. and Carnegie Group Inc. For this set, there are five content related types of categories: Exchange (39 categories), Organizations (56 categories), People (267 categories), Places (175

[2]In November 2000 Reuters released the first volume of a new large data collection of Reuters News stories for use in NLP and IR research. This first volume includes about $810,000$ English documents from August 20, 1996 to August 19, 1997. This collection has been used in the TREC-10 filtering track that took place from February to September 2001 while I was writing my dissertation.

categories), Topics (135 categories). The Topics categories represent economic subject categories. Examples include *coconut*, *gold*, *inventories*, and *money-supply*. This set of categories is the one that has been used in almost all the research with the Reuters-1987 data [2, 16, 43, 63, 72, 96, 130, 136]. Hayes *et al.* [43] used these 135 categories for evaluation, with only 723 documents in the test set and the remaining 21,450 documents in the training set. This version has been used in [43].

Version two was prepared by Lewis [72] and contains 21,450 documents. As noted by Lewis, the original test collection (Version 1) had been chosen without attention to certain properties of the data. In particular, there was some overlap between the training and test sets used in the CONSTRUE study because the same article was sometimes sent to Reuters in several slightly different versions on the same day. For this reason, the 723 documents from CONSTRUE's test set were set aside in Lewis' study. The remaining documents were split into two chronological groups. All the stories which originally appeared before April 7, 1987 were used for training (14,704 documents), and the stories from April 8, 1987 or later were in the test set (6,746 documents). This version has a set of 112 categories with at least one example in the training set. Of these categories 94 have one or more occurrences in the test set while 90 have one or more occurrence in both. The average number of categories per document is 0.64. 53% of the documents in the training set and 42% in the test set have more than one category, but many have none at all. This collection was used by [16, 63, 72, 78, 140].

Apte, Damerau & Weis [2] created version 3 of the Reuters-1987 collection by discarding from the Lewis' version 1 all those documents with "empty" topic assignments and selecting only those categories that have at least one example in the training set. This version has $10,645$ documents for training, $3,672$ documents for testing, and 93 categories. This collection has been used in [2, 140, 141].

The fourth version of the Reuters-1987 collection was build by Wiener *et al.* from the original Reuters-1987 collection [130]. They eliminated repeated stories, and selected those that have at least one assigned topic. The training and test set were selected by chronologically dividing the collection into many small chunks that do not overlap, numbering these chunks, and selecting the odd numbered chunks for training and the even numbered chunks for testing. This resulted in the collection being divided into $9,610$ documents for training and $3,662$ documents for testing. They use the 92 categories that have at least one example in the training collection for evaluation. This collection has been used in [129, 130, 140].

During the 1996 ACM SIGIR Conference, a group of researchers recognized the problems that all these variants had caused and the need to build a standard Reuters-1987 collection [75]. The goal was to fix several problems present in the original collection. Duplicate stories as well as unclassified stories (stories with no topic assignment) were marked explicitly and excluded from the splits defined. Such a collection was released at the end of 1997 and has been used for text categorization experiments since 1998 [25, 65, 118]. The new standard Reuters-21578 collection has

Table 2.1: Collections used for text categorization research

Collection (version)	Prepared by	# categ	# Train Doc.	# Test Doc.	Unused Docs.	partition (Training set)
Reuters-1987 (1)	CGI	135	21,450	723	0	random
Reuters-1987 (2)	Lewis	112	14,704	6,746	723	Chronological (up to 4/7/87)
Reuters-1987 (3)	Apte et al.	93	10,645	3,672	7,856	Chronological (up to 4/7/87)
Reuters-1987 (4)	Wiener et al.	92	9,610	3,662	7,856	Random chunks (odd numbered sets)
Reuters standard (Reuters-21578)	Lewis	135	13,625 (Lewis split)	6,188	1,765	Chronological (up to 4/7/87)
		135	9,603 (Apte split)	3,299	8,616	Chronological (up to 4/7/87)
OHSUMED	Lewis	119	183,229	50,216		Chronological (years 1987-1990)
AP TREC	Lewis	20	142,791	66,992		Chronological (years 1988-1989)

21,578 documents and includes five sets of categories topics described before. This standard collection also defines two training-testing splits: the modified-Lewis split and the modified-Apte split. These splits follow the training and testing data set definition of the previously mentioned studies, for example the Lewis split uses all the stories from April 7 and before for the training set while stories from April 8 and after for the test set. However, because of the elimination of duplicate stories and correction of many spelling errors in the categories the results of text categorization experiments conducted with this collection are not comparable with those conducted using the previous versions of the Reuters-1987 collection.

2.4.2 OHSUMED Collection

The OHSUMED test collection prepared by Hersh *et al.* was the first publicly available large collection for information retrieval research [46]. It was prepared by selecting all the articles published in 270 medical journals between 1987 and 1991.

The collection contains 348, 566 MEDLINE records. A MEDLINE record has several fields such as title (.T), abstract (.W), source (.S), author (.A), etc., (see Figure 2.1). These records are indexed with MeSH (Medical Subject Headings) categories (the .M field) that have been manually assigned by indexers at the National Library of Medicine (NLM). A subset of 233, 445 OHSUMED records have title, abstract and MeSH categories and was selected by Lewis *et al.* for text categorization research [78]. The records from the years 1987 to 1990 (183, 229 documents) were used for training, and the documents from the year 1991 (50, 216 documents) were used for testing. Relative to Reuters, OHSUMED has been used by fewer researchers to explore text categorization [65, 67, 78, 143]. The main reason seems to be the size of the collection and the large number of associated categories. The text categorization subset of OHSUMED has 14, 626 different categories in contrast to Reuters which has between 90 to 675 depending on what set of categories is used. A subset of MeSH, specifically the Heart Diseases subtree consisting of 119 categories, has been used by several researchers [65, 67, 78, 135] and is now a de facto standard comparison set. However, as with the Reuters collection, not all these studies agree on the test set used. For example Yang [135] reported results on the same subset of Heart Diseases categories but using only those documents that are positive examples of any of the 119 categories. This reduced the raining set to 12, 267 documents for training and 3, 267 documents for testing. Lam and Ho [65] use the same category subset but limit the documents to the year 1991 (50, 216 documents). Their test set includes only the last 16,738 documents of 1991 while the remaining 33,478 documents are used for training. Given the differences, comparisons between studies are to be made with caution.

The OHSUMED collection has also been used in the context of the TREC-9 filtering track [101]. The filtering track assumes that a stream of incoming documents has to be processed by the system to find documents relevant to the user's interests represented by profiles which reflect long term information needs. There are three tasks in TREC filtering: adaptive filtering, batch filtering and routing. In the adaptive filtering task the system starts with a user profile and a few positive examples (two or four). Each document that is retrieved is immediately judged for relevance and the system can use this information to change the profile. In batch filtering and routing the system has access to a large set of training documents that can be used for tuning the search profiles. In batch filtering the system must decide which documents should be marked as relevant[3], while in routing the system should return a ranked list of documents. For TREC-9 all the 348,566 documents of the OHSUMED collection were used. For batch and routing the documents of 1987 were used as training and the remaining four years as testing. A subset of 63 of the original OHSUMED queries was selected for filtering. In addition, a set of 4,903 MeSH headings were selected by choosing those headings that had at least 4 relevant documents and had at least one example in the last year of the test set[4]. Each MeSH topic forms a

[3]This is similar to the text categorization task that we have described before.

[4]Excluding MeSH headings not represented in the final year warranties that headings

corresponding profile and the documents assigned to the OHSUMED categories are regarded as relevance judgments for each topic. This set of $4,903$ profiles proved to be computationally demanding[5] and only 4 groups out of the 14 participants submitted results on this set. For this reason a smaller subset of 500 randomly selected profiles was defined to allow participation of all the groups in the track.

An important feature of the MeSH categories is that they are organized around a hierarchical structure. Interestingly this organization has been ignored in all text categorization studies published so far. In other words researchers have not tried to exploit the fact that some categories are more general or that others are highly specialized etc. Since our goal is to test the effectiveness of hierarchical text categorization methods, OHSUMED is the best suited collection for our experiments.

2.4.3 AP-TREC Collection

The AP TREC collection contains $242,918$ Associated Press news stories from 1988 to 1990. This collection has been used as part of the data set for the Text

that have been dropped out of MeSH (which undergoes continual modification) are not considered in the evaluation.

[5]Observe that if we use a classifier for each category that requires a total of 10 minutes to train and process the test set we would need about 34 days to process the entire set of 4903 classifiers on a single processor. Using multiple processors we could reduce this time significantly but this reduction is bounded by the amount of resources that can be dedicated to the task. During the TREC-9 conference most of the groups that completed this task said that they needed nearly 15 days to obtain a complete set of results.

```
.I 55402
.S
Heart Lung 8801; 16(5):584-9
.M
Adult; AIDS-Related Complex/*DI; Cardiac Tamponade/DI/ET;
Case Report; Electrocardiography; Endocarditis, Subacute
Bacterial/DI/ET; Female; Heart Diseases/DI/*ET/RA; Heart
Failure, Congestive/DI/ET; Heart Valve Diseases/DI/ET;
Human; Male; Myocardial Diseases/DI/ET; Support, Non-U.S.
Gov't.
.T
The Miami vices in the CCU. Part II. Cardiac manifestations
of AIDS.
.W
Cardiac manifestations of AIDS probably occur more frequently
than is appreciated --despite autopsy reports indicating that
more than 50\% of deceased AIDS patients had myocarditis.
A high index of suspicion and the echocardiogram will help in
revealing the true incidence of cardiac involvement in AIDS.
.A
Valle BK; Lemberg L.
```

Figure 2.1: Example record from OHSUMED

Retrieval Conference (TREC) for ad-hoc retrieval[6], routing and information filtering[7] tasks. Lewis *et al.* [78] selected a subset of $209,783$ AP stories that contain exactly one HEAD field (i.e., title) and TEXT field (i.e., the body of the article). A set of 20 categories was selected by combining two sets of 10 topics that had been used in several works previously published (10 topics used in [14, 16, 76], and another set of 10 topics used in [73]). Examples of these categories include tickertalk, boxoffice, bonds, gulf, Israel, and Japan. The documents from 1988 and 1989 ($142,791$ documents) are used for training and the year 1990 ($66,992$ documents) are used for testing. This collection has been used in [78, 118].

2.5 Evaluation Measures

A common strategy for evaluating the performance of machine learning methods is to use human performance as a gold standard. However, with text categorization a problem is that the manual assignment of categories to a given document is essentially a subjective task. Thus human indexers tend to disagree in their decisions. This well known phenomenon called *inter-indexer inconsistency* has been recognized in several studies (see for example [12, 42]). Nevertheless the performance of auto-

[6]The ad-hoc retrieval task of TREC is the classical IR task in which the document collection is known and relatively stable but the questions likely to be asked are not known.

[7]In a routing task the queries are known in advance while the documents are not known. This is similar to a situation in which a user has a topic to follow and receives a stream of news from which the system should rank documents according to the relevance to the user's interest. Routing can be seen as an inverted ad-hoc retrieval problem. Filtering is a more complex version of routing in which the system must perform a binary decision and report only those documents relevant to the user's interest.

Table 2.2: Contingency table for binary decision classification

	Class Positive ($C+$)	Class Negative ($C-$)
Assigned positive ($R+$)	a (True Positives)	b (False Positives)
Assigned negative ($R-$)	c (False Negatives)	d (True Negatives)

Table 2.3: Measures for binary classification defined in Information Retrieval (IR) and Artificial Intelligence (AI) communities

IR	AI	Formula
Recall (R)	*Sensitivity*	$\frac{a}{a+c}$
Precision (P)	*Predictive_value*$(+)$	$\frac{a}{a+b}$
Fallout	*Predictive_value*$(-)$	$\frac{b}{b+d}$
	Accuracy	$\frac{a+d}{a+b+c+d}$
	Specificity	$\frac{d}{b+d}$
Error_rate		$\frac{b+c}{a+b+c+d}$

matic text categorization is evaluated against results generated by human intellect.

In general, to evaluate a binary decision task we use a contingency matrix that represents all possible outcomes of the classification task (see Table 2.2). The information in this table underlines several evaluation measures. As Table 2.3 shows some of these measures are more commonly used in AI research while others are more dominant in IR.

None of these measures alone are appropriate to measure performance of text categorization algorithms. Recall (Sensitivity), and precision (Predictive value(+)) if used alone might show deceiving results, i.e. a system that assigns the category to every document (trivial acceptor) will show perfect recall (1.0). Accuracy, on the other hand works quite effectively if the number of positive and negative examples are balanced. However, it can also be deceiving in text categorization because the number of negative examples is typically overwhelming compared to the number of positive examples. In such situations, a system that assigns no documents to the category (trivial rejector) will obtain an accuracy value close to 1.

Measures that combine recall and precision have been defined in IR: break-even point, and F_β measure. Break-even point proposed by Lewis [72] and used in [63, 65, 72, 140] is defined as the point at which recall equals precision. van Rijsbergen's F_β measure [127] combines recall and precision into a single score according to the formula:

$$F_\beta = \frac{(\beta^2 + 1)P \times R}{\beta^2 P + R} \tag{2.1}$$

F_0 is the same as precision, F_∞ is the same as recall. Intermediate values between 0 and ∞ apply different weights assigned to recall, and precision. The most common values assigned to β to evaluate performance in IR are 0.5 (recall half as important as precision), 1.0 (recall and precision equally important), and 2.0 (recall twice as important as precision). The F measure has been used in [64, 78, 89, 143].

Break-even point (BEP) has some disadvantages. Usually the value of the break-even point has to be interpolated. Thus if the values of recall and precision are too far then BEP will show values that are not achievable by the system. Also the point where recall equals precision is not necessarily desirable nor informative from the user's perspective.

van Rijsbergen's F measure is the most suited measure, but still with the drawback that it might be difficult for the user to define the relative importance of recall and precision. In this dissertation we will report results using F_1 values because it will allow us to compare results with other works that use the same dataset [78, 143].

In general the F_1 performance is reported as an average value. There are two ways for computing this average: macro average and micro average. With macro average the category perspective is used, therefore the method computes the F_1 values for each category and averages them to get the final macro averaged F_1. Another approach is to use the document perspective. In this case each categorization decision is equally important. We first obtain the global values for the true positive, true negative, false positive, and false negative scores and then compute the micro averaged F_1 value using the recall and precision computed with the global values.

Two other measures have been used in the context of TREC evaluation for measuring performance of filtering systems T9P and T9U. T9P is a precision oriented measure defined as:

$$T9P = \frac{R_+}{Max(Target, (R_+ + N_+))}$$

where R_+ is the number of relevant documents retrieved, N_+ is the number of non relevant and $Target = 50$ documents. T9U is a linear utility function that gives a credit of 2 for every relevant document retrieved and a penalty of 1 for a non-relevant document retrieved. This is equivalent to filtering documents with estimated probability of relevance above 0.33. Because the scores of the utility function can be unbounded, T9U also defines a lower bound for the minimum score that can be applied to a topic. The formula for $T9U$ is:

$$T9U = Max(2 \times R_+ - N_+, MinU)$$

where $MinU$ is a minimum negative value defined as -100 for the OHSUMED topics and -400 for the MeSH topics.

2.6 Applications of Text Categorization

As we stated in Chapter 1, classification is a core task in many human activities. For this reason we can find applications of text categorization in many practical problems. In this section we will describe some of the most common applications for text categorization.

2.6.1 Document Indexing

Document indexing has been traditionally a manual task performed by librarians to facilitate access to the books in the library. However, it also can be automatic. Automatic indexing, proposed in the early 1960's in Maron's [81] seminal work, is the application that motivated most of the early research in text categorization [6, 31, 32, 40, 41, 42, 44, 47, 62, 81]. In these early studies the goal was to perform automatic indexing to improve retrieval performance in Boolean information retrieval systems. In these systems each document is assigned one or more keywords that describe the content of the document. These keywords are terms from a controlled vocabulary such as the MeSH or ERIC thesaurus. The motivation for automatic indexing research derived from the fact that manual indexing performed by trained human indexers is an extremely costly activity. In this application the entries from the thesaurus are viewed as categories, and document indexing becomes a multi-label text categorization problem. Several automatic document classifiers have been described in the literature [4, 36, 38, 102, 125].

Automatic indexing is closely related to the emerging domain of *automatic metadata generation* [120]. This application is particularly useful in digital libraries where we are interested in tagging documents with metadata representing several aspects, for example creation date, author, document format, etc. Metadata such as bibliographic codes, or keywords may also describe the semantic of the document. The generation of metadata can be accomplished by using automatic text categorization.

Another example of this kind of application is the classification of patents. Every year the U.S. Patent and Trademark Office as well as the European Patent Office receive an increasing number of patent applications that must be channeled to the appropriate experts for study leading to a decision about whether to grant the patent or not. A recent paper published by Larkey [70] shows a solution using text categorization. Observe that this application can be viewed as a special case of document indexing in which each patent must be assigned to a single category.

Automatic text categorization has recently been used to index and organize information on the World Wide Web. It has been used to organize results from a search engine, which is a method used by Northern Light and Google, with the goal of facilitating the browsing of large numbers of documents. The WebKB project at Carnegie Mellon University uses text categorization to classify web pages into hierarchies such as Yahoo! topics [20, 21, 58, 84, 100].

2.6.2 Document Filtering

Document filtering refers to the task of classifying a dynamic collection of documents, for example a stream of incoming news wire stories. Filtering may be viewed as a single-label categorization problem. Given a description of a topic (generally called a *profile*), an information filtering system checks whether or not an incoming news article is relevant to the topic. Additionally, a filtering system may perform a further categorization step by organizing the relevant documents into topical categories. The described framework could also be used to filter and organize e-mail [16].

As mentioned before the Text Retrieval Conference (TREC) has a research track dedicated to the study and evaluation of filtering strategies. The most popular task is that of *adaptive filtering* [50]. In this task the filtering system starts with an initial profile (derived from very few documents) and receives feedback from the user on the relevance of documents as they are presented to the user. The goal of the system is to learn the characteristics of those documents deemed relevant by the user and use them to adapt the user profile. A similar categorization task is studied within the Topic Detection and Tracking (TDT) conference but here the incoming stream is of audio format or the close caption text from TV/radio broadcasts news. The TREC filtering track, as well as the topic detection task assume that a small set of positive examples (usually 4 examples) is given to the system. The goal is to track the development of the event [137] in the stream of news received by the system.

Text categorization has been applied to filter unsolicited bulk e-mail, also known as *spam*. Spam messages are annoying to most users because of the waste of space and time invested in deleting them. A 1997 study found that spam constitutes approximately 10% of the incoming messages to a corporate network [19]. Two studies have been published regarding the application of text categorization to build anti-spam filters [1, 112]. These works evaluate the performance of naive Bayesian classifiers as anti-spaming filters. One would expect that filtering spam messages using keywords will not work very well because in general spam messages do not have an actual content or domain (many legitimate messages share the vocabulary of spam messages). However both studies have showed that the naive Bayesian classifiers are quite effective for filtering spam messages obtaining between 85 to 88% recall and 95 to 97% precision.

2.6.3 Other Applications

Word sense disambiguation is one of the most challenging problems in computational linguistics. It can be defined as the task of finding the “correct” sense that an ambiguous (i.e. a polysemous or homonymic) word has in a text. For example, the word **bank** has at least two different senses, as in **the Bank of America** or **the bank of the Mississippi river**. Word sense disambiguation can be viewed as a text categorization problem in which the context of the word occurrence is the document and the different senses are the categories. Several works in the literature [39, 45] have used text categorization strategies to solve the problem of word sense disambiguation.

The most unusual application of text categorization that we have found is an automatic essay grading system presented by Larkey [69]. In this application several classifiers were trained to assign scores using a training set of manually-graded essays. These scores were combined with several other summary text measures using linear regression. When tested on a new set of essays the agreement between the automated grader and the final manual grades was as good as the agreement between human graders.

In this chapter we provided a general review of supervised learning algorithms for text categorization with particular emphasis on document representation. We also described evaluation measures and some of the recognized variations across studies in using the standard test collections. Finally we illustrated some of the application contexts that embed text categorization research.

CHAPTER 3
TEXT CATEGORIZATION METHODS

This chapter presents a survey of the supervised machine learning methods that have been used in text categorization. Different approaches based upon Decision trees (ID3) [37, 77, 92], inductive rule learning [2, 14, 15, 17], neural networks [105, 119, 130], Linear Classifiers [5, 49, 78], K-nearest-neighbor (KNN) algorithms [22, 56, 67, 134, 135], support vector machines (SVM) [26, 54, 56, 136], and Naive Bayes [63, 77, 81, 84] have been explored. We will show that most of the research in text categorization has focused on building classifiers without regard to the hierarchical structure of the classification scheme. Only recently some works [63, 84, 90, 96] have tried to take advantage of the hierarchical structure in certain classification schemes.

Automatic text categorization approaches can be classified according to their theoretical foundations into:

- Classical IR based classifiers
- Statistical learning classifiers
- Linear classifiers
- Instance-based classifiers
- Decision trees

- Inductive rule learning
- Expert systems
- Neural networks
- Support vector machines

In general, the text categorization problem is framed as a supervised learning task where the classifier learns the unknown target classification function from a training set of examples. The classifier is then assessed by applying it to new instances in a test set.

The following sections will describe some of these approaches and present their major results. We will use the following notation (as specified in chapter 2): $C = \{c_1, \ldots, c_n\}$ is the set of categories and $D = \{\mathbf{d}_1, \ldots, \mathbf{d}_N\}$is the set of documents, $\mathbf{d} = (x_1, \ldots x_m)$ is the vector that represent the document in the m dimensional term space.

3.1 Classical IR Based Classifiers

This group includes several algorithms for categorization such as Rocchio's algorithm [103], and retrieval-threshold based algorithms (WORD) [135, 136]. The retrieval threshold algorithm is a simple classifier that learns by ranking all the documents of the training collection according to the similarity to the name and/or the description of the category and selecting an optimal similarity threshold that maximizes the performance measure. The learned classifier is then applied to the test set where all documents above the threshold are categorized as positive instances of the category. The retrieval threshold algorithm usually performs poorly and in some studies has been used as a baseline value [136].

Rocchio's algorithm was developed in the mid 60's to improve a query using relevance feedback. It has proved to be one of the most successful relevance feedback algorithms. Rocchio [103] showed that the *optimal query* vector is the difference vector of the centroid vectors for the relevant and the non-relevant documents. Salton & Buckley [114] included the original query in the computation of the optimal query so as to retain the focus of the query. They also added coefficients to control the contribution of each component as shown below:

$$\vec{Q}_{new} = \alpha\vec{Q}_{orig} + \beta\frac{1}{R}\sum_{d_j \in rel}\mathbf{d_j} - \gamma\frac{1}{N-R}\sum_{d_j \notin rel}\mathbf{d_j} \tag{3.1}$$

where $\mathbf{d_j}$ is the weighted document vector, $R = |rel|$ is the number of relevant documents, and N is the total number of documents. All the negative components of the final vector $\vec{Q}_{new}$ are set to zero. Several techniques have been proposed to improve the effectiveness of Rocchio's method: better weighting schemes [122], query zoning [123], and dynamic feedback optimization [10].

For text categorization the first term of Rocchio's formula is zero because we do not have an initial query. The other two terms represent the centroid of the positive examples, and negative examples respectively and the classifier is the vector of the difference between these centroids. For a given category $c_i \in C$ we compute a vector **w** (which represents the classifier) using the formula:

$$\mathbf{w_i} = \beta \frac{1}{R} \sum_{d_j \in c_i} \mathbf{d_j} - \gamma \frac{1}{N-R} \sum_{d_j \notin c_i} \mathbf{d_j} \tag{3.2}$$

Classification of a new document **d** consist in computing $f(\mathbf{d}) = \mathbf{w_i} \cdot \mathbf{d}$. For binary classification a threshold t is selected (usually by optimizing a performance metric such as F_1) using the training set and the category c_i is assigned if $f(\mathbf{d}) > t$.

Text categorization based on Rocchio's algorithm was first proposed by Hull [49] and since then it has been used as a base line in most text categorization studies [17, 54, 65, 78, 118, 119, 134, 140]. Several researchers have explored Rocchio's algorithm specifically to study its characteristics and behavior for categorization [53, 99, 109]. Its performance is highly dependent on document the weighting scheme but when properly tuned it is usually equivalent to other methods that are more complex [118]. We will explore this classifier in more detail later in this dissertation and compare its performance with other classifiers.

3.2 Statistical Learning Classifiers

These are classifiers that are based on statistical learning methods. The first text classifier reported in the literature used a probabilistic approach [81]. We present two members of this group: Bayesian learning and regression models for text categorization.

The most common statistical method for text categorization is the naive Bayes classifier [63, 77, 84, 88]. In general, the Bayesian approach to classify a new instance is to assign the most probable target value, given the set of attributes $A = (a_1, \ldots, a_m)$ that describe the instance and the set of classes $C = (c_1, \ldots, c_n)$:

$$v_{NB} = argmax_{c_j \in C} P(c_j) \prod_i P(a_i | c_j) \tag{3.3}$$

where v_{NB} is the target value output by the naive Bayes classifier, and $a_i \in A$. In the learning step the different terms $P(c_j)$ and $P(a_i|c_j)$ are estimated based on their frequencies in the training set. The set of these estimates is the learned hypothesis. This hypothesis is then used to classify a new instance using formula 3.3. This formula is obtained under the assumption that the attribute values are conditionally independent given the target value.

In text categorization the attributes are the terms (stems and/or phrases) in the text that we want to classify. The naive Bayes classifier maximizes the probability of observing the terms that are in the text, subject to the Bayes independence assumption. Observe that the independence assumption is often violated in natural language. For example, the probability of observing the word *"intelligence"* is greater if it is preceding by the word *"artificial"*. Despite the violation of the independence assumption the naive Bayes classifier has performed very well in text categorization of web pages [84, 85, 88, 91]. Bayesian classifiers have been tested in several standard collections including Reuters (versions 2 and 3) [77, 92], news groups postings [84, 85], and web pages [84, 85, 88, 91]. The results on the Reuters collection are 0.65 and 0.71 micro-averaged BEP on versions 2 [77] and 3 [92] respectively. A comparative study by Yang shows that the naive Bayesian classifier has the lowest results of all the learning methods reported in the Reuters collection.

Regression is another statistical learning method that has been used in text categorization [52, 76, 119, 139]. Regression refers to the problem of approximating a real-valued function f by means of a function $\hat{y}$ that fits the training data [88, 120]. The most successful regression method for text categorization is the Linear Least Square Fit (LLSF) method which was proposed by Yang and Chute [138, 139]. In LLSF a multivariate regression model is automatically learned from the training set of documents and categories. Each document d_j has two vectors associated with it: an input vector $I(d_j)$ (i.e., the standard document vector of m components one for each term) and an output vector $O(d_j)$ which defines the weight of the n possible classes. A matrix ($\hat{M}$) of linear regression coefficients is obtained by solving a linear least square fit on the training pairs of vectors. This matrix defines a mapping from the words of an arbitrary document to the set of categories. The categorization of a new instance is performed by multiplying the regression matrix by the document's input vector, obtaining a ranked list of categories that can be assigned ($\hat{M}I(d_j) = O(d_j)$). LLSF computes the matrix $\hat{M}$ from the training set by computing the linear least square fit that minimizes the error of the training set according to the formula:

$$\hat{M} = argmin_M ||MA - B||_F \tag{3.4}$$

where $||V||_F = \sum_{i=1}^{r} \sum_{j=1}^{s} v_{ij}$ represents the *Frobenius norm* of an $r \times s$ matrix, A is the $m \times N$ matrix whose columns are the input vectors of the training documents, and B is the $n \times N$ matrix whose columns are the output vectors of the training documents. M is an $n \times m$ matrix obtained by performing singular decomposition on the training set.

LLSF is one of the most effective classifiers reported in the literature. It has been tested using the Reuters collection (Version 3) obtaining 0.85 micro-average BEP [140]. Despite its success in relatively small and medium size sets of categories, the high computational cost associated with the computation of singular decomposition makes it difficult to use LLSF in large dimensional problems.

3.3 Linear Classifiers

Linear classifier algorithms use a vector to represent a category in the same dimensional space as the documents and a linear function $f(\mathbf{d})$ to estimate the similarity between the category and the document. Documents with the largest values of $f(\mathbf{d})$ are likely to be members of the class. The linear function may be expressed as:

$$f(\mathbf{d}) = \mathbf{w} \cdot \mathbf{d} = \sum_{j=1}^{m} w_j x_j \tag{3.5}$$

where $\mathbf{d} = (x_1, x_2, \ldots, x_m)$ is the document vector that represents the text, $\mathbf{w} = (w_1, w_2, \ldots, w_m)$ is the vector that represents the classifier, and m is the number of features that represent a document from the collection.

Linear classifiers are also called *profile-based classifiers* because they rely on the extraction of an explicit *profile* from the training set to represent a category [120].

The Rocchio classifier is an example of a linear classifier that presents the profile of a class as the difference between the centroids of its positive and negative examples.

Several linear classifiers have been used for text categorization: linear discriminant analysis [5, 48, 119], perceptron algorithm [96, 119, 130], Widrow-Hoff algorithm [65, 78], and Kivinen & Warmuth's EG algorithm [78].

Linear discriminant analysis is based on finding the linear combination of the variables that maximizes the separation between categories. The method characterizes each class by its estimated mean vector ($\overline{\mathbf{d}}_i$) and covariance matrix (S_i) measured over the training set. The mean vector and covariance matrix are defined as:

$$\overline{\mathbf{d}}_i = \sum_{d_j \in c_i} \frac{d_j}{|c_i|} \tag{3.6}$$

$$S_i = \frac{1}{|c_i| - 1} \sum_{d_j \in c_i} (\mathbf{d}_j - \overline{\mathbf{d}}_i)(\mathbf{d}_j - \overline{\mathbf{d}}_i)^T \tag{3.7}$$

where $|c_i|$ is the number of documents that contain this class. A new document $\mathbf{d}$ is classified into the class with the nearest mean vector, scaled for the shape of the covariance matrix, based on the Mahalanobis distance metric:

$$dist(\mathbf{d}) = (\mathbf{d} - \overline{\mathbf{d}}_i)^T S_i^{-1} (\mathbf{d} - \overline{\mathbf{d}}_i) \tag{3.8}$$

This method has been successfully used in routing[1] by Hull [49]. Blosseville *et*

[1]To remind the reader, in routing the classifier has to return a ranked list of documents related to a given topic.

al. used linear discriminant classifiers to classify projects into disjoint categories [5].

The Widrow-Hoff algorithm (WH) runs through the training set for each category one example at a time updating the weight vector at each step. Initially the vector is set to zeros. At each step the new weight vector $\mathbf{w_{i+1}}$ is computed from the old weight vector using the training example $\mathbf{x_i}$ with label y_i. The jth component of the new weight vector is computed as:

$$w_{i+1,j} = w_{i,j} - 2\eta(\mathbf{w_i} \cdot \mathbf{x_i} - y_i)x_{i,j} \tag{3.9}$$

The parameter $\eta > 0$, which is usually called the *learning rate*, controls the extent to which $\mathbf{w}$ is allowed to change and how much influence each new example has on it. Observe that the second term of the formula $2(\mathbf{w} \cdot \mathbf{x} - y)\mathbf{x}$ is the gradient with respect to $\mathbf{w}$ of the square loss $(\mathbf{w} \cdot \mathbf{x} - y)^2$. Thus the WH algorithm tries to move in the direction that maximizes the loss.

The Kivinen & Warmuth's *exponentiated-gradient* algorithm (EG) is similar to WH in that it also uses a weight vector $\mathbf{w_i}$ for each category and runs through the training set one example at a time. The EG differs from WH in that all the components of the weighting vector are nonnegative and must sum to one. Usually the initial vector is set to $\mathbf{w_1} = (1/m, \ldots, 1/m)$. The EG rule for updating weights with each new example is:

$$w_{i+1,j} = \frac{w_{i,j}\exp(-2\eta(\mathbf{w_i}\cdot\mathbf{x_i} - y_i)x_{i,j})}{\sum_{j=1}^{d} w_{i,j}\exp(-2\eta(\mathbf{w_i}\cdot\mathbf{x_i} - y_i)x_{i,j})} \tag{3.10}$$

Observe that each component $w_{i,j}$ is multiplied by $\exp(-2\eta(\mathbf{w_i}\cdot\mathbf{x_i}-y_i)x_{i,j})$, and then the entire vector is renormalized. As before, the learning rate $\eta > 0$ controls the impact of the new training examples. According to Kivinen & Warmuth each new weight vector $\mathbf{w_{i+1}}$ can be shown to maximize a formula which trades off two conflicting goals: (i) maximizing the loss $(\mathbf{w_{i+1}}\cdot\mathbf{x_i} - y_i)^2$ of the vector $\mathbf{w_{i+1}}$ on the current example $\mathbf{x_i}$, and (ii) penalizing a new vector $\mathbf{w_{i+1}}$ which is “too far” from the old vector $\mathbf{w_i}$. The different rules of EG and WH are derived using different choices of distance functions in (ii). The parameter η determines the relative importance given to goals (i) and (ii).

WH and EG algorithms have been tested on the AP collection, and the OHSUMED collection by Lewis *et al.*[78]. They report that WH and EG achieve F_1 performance values of 0.55 and 0.50 respectively which is significantly better than the 0.44 obtained by them using a simple non optimized version of Rocchio classifier.

3.4 Instance-Based Classifiers

Instance-based methods such as nearest neighbor and locally weighted regression are learning algorithms that simply store the presented training data, in contrast to methods that construct a general, explicit description of the target function when training examples are provided. Generalizing beyond these examples is postponed until a new instance is classified. Every time a new document needs to be classified, its relationship to the previously stored examples is examined in order to assign the target categories for the new document. Instance-based methods are also known as *lazy* learning methods because they delay processing until a new instance must be classified [88, 120].

Described in the mid 1960's by Cover & Hart [18], the K-Nearest Neighbor (KNN) algorithm is one of the most basic instance-based methods for pattern recognition. Given an object represented in the multi-dimensional term space, its k-nearest neighbors are defined in terms of the standard Euclidean distance. The application of the KNN algorithm to text categorization was initially proposed by Creecy and Masand in the early 1990's [22, 82]. They used a KNN based classifier to categorize stories for the Dow Jones in a CM-5 machine. A weighted-distance version of KNN for text categorization has been used by Yang obtaining very good results in the Reuters collection and in the OHSUMED collection [135, 136, 141]. The categorization of a new document in the test set is based on the categories assigned to the k closest documents from the training collection. Given a new document from the test set, the algorithm finds the k nearest documents from the training collection and ranks them by similarity value. The similarity of each neighbor to the new document is used to weight each category of the neighbor, and the sum of category weights over the k nearest neighbors are used for category ranking. Those categories above a threshold are assigned to the document. Observe that in contrast to other methods that perform a binary assignment for each single class, such as Bayesian classifiers, the KNN algorithm can assign multiple categories to a document. Despite its simplicity, KNN has proven to be one of the most effective algorithms for text categorization. Yang [136] has reported results on the Reuters collection versions 2, 3, and 4 obtaining micro-averaged BEP of 0.69, 0.85, and 0.82 respectively. These values are comparable to the performance achieved by neural networks, and LLSF.

3.5 Decision Trees

Decision trees have also been applied to text categorization by several authors [77, 92, 93]. Lewis and Ringuette [77] used the IND package developed by Buntine [11] to implement a decision tree (DT-min-10) for the Reuters (version 3) collection. For each category they build a decision tree using the recursive partitioning algorithm with information gain split rule. A leaf is forced whenever a node has less than 10 examples. No pruning was done on the final tree. Lewis and Ringuette report a micro-averaged BEP of 0.67 which is a low performance for the Reuters collection (version 2).

Moulinier [92] uses an implementation of ID3 with the Reuters collection (version 3) reporting a good performance (micro-averaged F_1=0.78).

The most significant application of decision trees to text categorization was done in the AIR/X project developed at Dortmund University [4, 36, 37, 38]. This was a project spanning more than 10 years that produced an operative system for the classification of scientific literature. The text categorization approach employed in AIR/X is known as the *Darmstadt Indexing Approach* (DIA). The DIA is based on the computation of association factors called $z(t_k, c_i)$ between free text terms t_k and categories c_i, where $z(t_k, c_i) = \frac{P(t_k,c_i)}{P(t_k)}$. This is simply the conditional probability $P(c_i|t_k)$, i.e. the portion of training documents containing t_k that are classified under c_i. After computing these factors, the DIA trains a decision tree in two steps. In the first step (called the description step), for every occurrence t^x_{kj} of the term t_k in training document d_j, a relevance description vector $\mathbf{rd}(c_i, d_j)$ is updated by using $z(t_k, c_i)$ and the characteristics of occurrence t^x_{kj} (i.e. the section of d_j that contains t^x_{kj}). In the second step (called decision step), the relevance description $\mathbf{rd}(c_i, d_j)$ is transformed into a discrete-valued vector $\mathbf{rd}(c_i, d_j)$. At this point, the ID3 decision tree algorithm is invoked. The ID3 algorithm selects one attribute of the vector representation at a time using a χ^2 criterion to partition the training vectors into equivalence classes of identical vectors. to categorize a new test document $\mathbf{d}$ the decision and description steps are applied to $\mathbf{d}$. The percentage of training vectors $rd(c_i, d_j)$ assigned to the equivalence class of $rd(c_i, \mathbf{d})$ (which corresponds to the probability that the assignment is correct) determines the final assignment assignment of categories to the document $\mathbf{d}$.

3.6 Inductive Rule learning

This group includes algorithms that automatically generate a set of rules for text categorization [2, 13, 15, 17, 93, 94]. This is an attractive solution because the set of rules can be interpreted directly by human indexers thereby supporting validity checks.

Apte, Damerau & Weiss used a technique called Swap-1 to generate induction rules for classifying the Reuters collection (version 3)[2]. This rule induction method attempts to find a "compact" covering rule set that completely partitions the examples into their correct classes. The set of rules is found by heuristically searching for a single best rule that covers cases in a class. This rule is the added to the set of existing rules, and the covered cases are removed from further consideration. The process is repeated until all the cases are covered. Once a covering set is found for all the categories, it is refined by pruning or statistical techniques. Using training and test evaluation methods, the initial covering rule set is then scaled back to the most statistically accurate subset of rules. For each category a set of features is selected from a *local dictionary* build from the words of all the positive examples of the category. The words in the local dictionary are ranked by document frequency and the top few are used as features by Swap-1.

RIPPER [16] uses a similar approach by repeatedly adding rules to an empty rule set until all positive examples are covered. Rules are formed by splitting the training set in two subsets a "growing set " and a "pruning set", and greedily adding conditions to the antecedent of a rule (starting with an empty antecedent) until no negative examples are covered; after such a rule is found, the rule is simplified by greedily deleting conditions so as to improve the rule's performance on the "pruning set". After covering all the positive examples, an optimization phase modifies the rule set and improves its fit on the whole training set.

RIPPER and Swap-1 methods have been tested on the Reuters collection (version 3) and yielded similar performance (0.79 and 0.80 micro-averaged BEP) [2, 16]. RIPPER has also been tested in the new Reuters standard collection (Reuters-21578) obtaining 0.696 in the ModeLewis split and 0.820 in the ModApter split [17].

3.7 Expert Systems

During the late 80's several papers reported expert systems for text categorization such as CONSTRUE [43] and MedIndex [51]. In general, these systems rely on a set of manually constructed rules, which is stored in a knowledge based, and one or more inference mechanisms that use the rules stored in the knowledge base to infer the correct classification of a document presented to the system. MedIndex is an expert system built at the National Library of Medicine for computer assisted document indexing of medical literature using MeSH headings [51]. The system uses manually constructed rules to select a set of candidate MeSH categories that could be assigned to a document. It is not a fully automated text categorization system because the final categorization decision is taken by human indexers who can expand or reduce the proposed set of categories. However, it is an example of a set of carefully constructed rules for text categorization. The CONSTRUE system is an expert system built specifically for text categorization of Reuters articles. It was the first work that used the Reuters collection (version 1). Hayes *et al.* report that it took about 1.5 person-years to develop the rules for text categorization in CONSTRUE [43]. The system shows an impressive performance (0.90 micro-averaged BEP) over a small subset of about 3% of the Reuters collection. Despite their initial success, expert systems for text categorization have not been further developed possibly because the adaptation to other domains as well as the expansion of the system to use new categories have proven to be extremely costly and labor intensive.

3.8 Neural Networks

Neural networks are machine learning methods that provide a robust approach to approximating real-valued, discrete-valued, and vector-valued functions [88, 107]. Neural networks are inspired by the observation of biological organisms and their large interconnected webs of neurons. They are a rough approximation of biological organisms in the sense that they are built out of small units (neurons) that are interconnected to form a web. Each neuron receives one or more real-valued inputs and produces a single real-valued output, which may become the input of another unit. Despite the similarity, there are many complexities in the biological systems that are not captured by the neural network models. The most successful method for supervised learning in neural networks is the backpropagation algorithm which was introduced by Rumelhart and his collaborators [107, 108].

Neural networks were introduced to text categorization in 1995 [119, 130] and several studies have used them since then [23, 96, 105, 106, 129, 140]. A neural network classifier consist of two or more layers of interconnected units. The input units represent the terms of the document, while the output units typically represent the categories that can be assigned to the document. The units are interconnected and the weights associated to the connection edges represent conditional dependence relations. Since this dissertation proposes the use of neural network classifiers using the hierarchical structure of the vocabulary, we will describe them in detail in subsequent chapters.

Wiener, Pedersen & Weigend [130] use a backpropagation neural network per category using two kinds of architectures: a flat architecture, and a modular architecture. The flat neural networks are backpropagation neural networks trained for each category on the entire training set. They use simple neural networks with a single hidden layer of six logistic sigmoid units. Their modular neural network architecture has two levels. The first level represents meta-topics while the second level represents specific sub-topics. The first level is a network trained on the full training set to estimate the probability that each of the five meta-topics (agriculture, energy, foreign exchange, government, and metals) is present in the document. A meta-topic is present if any of its subtopics is present. The second level contains specific topics. Each topic is represented by a neural network which is trained only on the documents that are positive examples of the meta-topic. The meta-topic network use fifteen hidden units and five outputs (corresponding to the five meta-topics). The topic networks use six hidden units and a single output. A document is classified by presenting it to the meta-topic network and to each of the topic networks. The outputs of the meta-topic network are then multiplied by the output of their corresponding topic networks to obtain a final estimate. The performance reported on Reuters collection (version 4) is 0.82 micro-averaged BEP which is one of the best results reported for this collection.

3.9 Support Vector Machines

Support vector machines for text categorization have been recently proposed by Joachims [54, 56, 57] and subsequently used by other researches [26, 124, 136]. In geometrical terms, this method can be seen as the attempt to find the best surface σ_i that separates the positive and negative examples. "Best" here means that σ_i separates the positive and negative examples by the widest margin. This method is an application of the *structural risk maximization principle*, according to which the decision surface should minimize the true error, i.e., the probability of misclassification of randomly selected, yet unseen test examples. The best decision surface is determined by only a small set of training examples called the *support vectors*.

Support vector machines offer important advantages for text categorization [54]:

- There is no need for term selection because support vector machines do not suffer from over-fitting and can scale up to considerably large dimensions.
- There is no need for parameter tuning on a validation set because there is a theoretically motivated "default" choice of parameter settings which has also been shown to provide the best effectiveness.

Joachims [55, 57] has published results for the Reuters collection and for the

OHSUMED collection using support vector machines. The SVM performance on new standard Reuters collection (Reuters-21578) using the ModApte split is 0.864 micro-averaged break-even point, which is the best results published for this collection. For the OHSUMED collection Joachims uses the 23 "Disease" categories of the MeSH classification and the first $20,000$ documents of the year 1991 dividing it into two sets of $10,000$ documents each that are used for training and testing respectively. He reports a macro-averaged break-even point of 0.660 on the 23 "Disease" categories. We have to note that the text categorization task is very different from the one described in any of the previously discussed works. Joachims uses only the high level disease categories and assumes that if a descendant in the UMLS tree is assigned then the general category is present. This assumption simplifies the problem to a general level of categorization and might be the reason for the high performance values reported in his SVM experiments, because general categories are easier to tell apart while more specific categories tend to be harder to separate. This assumption as well as the training-test split used prevents comparison of Joachims' results with any of the previously published results. Dumais *et al.* [26] have shown that support vector machine achieve training speeds comparable to computationally easy methods such as Rocchio. Recently, Dumais and Chen have explored the use of SVM for classifying very heterogeneous web content [25].

3.10 Hierarchical Approaches

One structural characteristic of many classification schemes is the presence of a hierarchical framework. In such schemes the more general categories (at high levels) typically lead to more specific categories (at lower levels). Interestingly only four of the works presented in previous sections explicitly use the hierarchical structure of the corresponding classification scheme [63, 84, 90, 96]. Three of them Mladenić [90, 91], Koller & Sahami [63, 111], and McCallum *et al.* [84] use Bayesian classifiers, while Ng, Goh, & Low [96] use perceptrons.

Koller & Sahami [63, 111] proposed a hierarchical approach that trains independent Bayesian classifiers for each node of the classification hierarchy. They use the Reuters collection and three simple hierarchies that are built by using as high level nodes those categories that tend to subsume other categories. The classification method starts at the root and selects the best link to a second level classifier. This process is repeated until a leaf is reached or until it is determined that none of the children is a good candidate. The system assigns the categories of the activated nodes. According to the authors, the hierarchical structure is used as a filter that activates only the best classification path. Observe that errors in classification at the higher levels are irrecoverable in the lower levels. Koller and Sahami report classification accuracy on each of the three small hierarchies and compare performance against a naive Bayes classifier. The best classification results for the first two hierarchies do not show significant differences between the hierarchical and flat classifiers (94.1 for both classifiers in Hier1, 90.0% accuracy for the hierarchical vs 87.7% accuracy for the flat classifier on Hier2). The third hierarchy shows a significant difference in accuracy between the hierarchical (98.6%) and the flat (95.7%) classifier.

Mitchell and his collaborators in CMU have been working on text categorization specifically targeting the problem of classifying web pages and the problem of extracting knowledge from the World Wide Web [20, 21, 58, 84, 100]. Their approach is also based on Bayesian classifiers. They use the hierarchical structure (such as the Yahoo! Hierarchy, and the Industry Sector Hierarchy) to improve the accuracy of Bayesian classifiers using a statistical technique called shrinkage that smoothes parameter estimates of a child node with its parent in order to obtain more robust estimates. The Bayesian classification schemes consists in estimating the parameters of the model from the training collection, and then applying the shrinkage method to improve these estimates using the predefined vocabulary hierarchy. The classification of the test set is performed by computing the posterior probability of each class given the words observed in the test document, and selecting the class with the highest probability. The performance of the flat classifier on the Yahoo! categories is 36.4% for the flat classifier and 39.5% for the hierarchical classifier. Their experiments show that shrinkage improves the performance when the training data is sparse, reducing the classification error by up to 29%.

Mladenić [90, 91] also explored hierarchical classification structures using the Yahoo hierarchy to classify web pages. Her approach also uses a Bayesian classifier.

For each node in the Yahoo subject hierarchy a classifier is induced. To train each of the non-leaf classifiers a set of positive examples is defined as consisting of all the positive examples of the node plus the positives examples of the descendants. These examples are weighted according to their position in the tree. The classification process on the test set works as described before for the Bayesian classifiers but only the set of categories with predicted probability ≥ 0.95 are assigned. The results reported on three domains (sub-trees) of the Yahoo! hierarchy are 0.33, 0.44 and 0.42 F_2.

Ng *et al.* [96] build their hierarchical classifier using perceptrons. Each node of the hierarchical classification tree is represented by a perceptron. They distinguish two types of nodes, leaf nodes and non-leaf nodes. They apply this to the Reuters corpus (version 3) where the categories reflect a 3-level geographical/topical hierarchy. The root node of their hierarchy is connected at the first level to nodes representing all the possible countries, and for each country different topics are defined, i.e. economics and politics. The leaf nodes are specific categories of the second level (e.g., for economics they have communications, industry, etc.). The hierarchical classifier receives a document and checks whether it belongs to any of the first level nodes (any of the different countries). If the tested document belongs to a country according to the classifier built for that country category, then the system checks for membership in the categories of the subtree rooted at that country category. If at any of the non-leaf nodes the process finds that none of its children is a good candidate, then the categorization stops at that branch of the recursion. The output of the classifier is the final set of leaf nodes reached in the recursion (zero, one or more). Observe that the system makes the underlying assumption that the categories in the path of the leaf node are also assigned. This methods is similar to the *Pachinko machine* proposed by Koller and Sahami, but with multiple outputs instead of a single output [63].

In general all of these studies have found improvements when the hierarchical structure information is used for building the classifiers. However, Koller & Sahami report some mixed results where, depending on the hierarchy used, the flat classifier outperforms the hierarchical classifier. This might be explained by the way they performed the binary decisions in the hierarchical approach, since with the high level nodes they select only the most probable branch. The errors at internal nodes compound, and as a consequence the incorrect decision cannot be recovered on the lower level nodes. In their approach the intermediate nodes must have a very high accuracy in order to obtain a performance higher than the flat classifier.

Unfortunately, the results of the four studies are not comparable because they use three different test collections (Yahoo! web pages, Reuters (version 3), and news group articles). Furthermore although Koller & Sahami, and Ng, Goh, & Low use the same Reuters collection (version 3), the results reported with the hierarchical classifier are not obtained for the same set of categories. Moreover the two studies report results with different performance measures (accuracy, and BEP respectively) which are not comparable.

Theoretically we observe that as shown by Mitchell [87] under certain conditions a hierarchical classifier like the *Pachinko machine* is equivalent to the non hierarchical classifier. His proof makes three assumptions: (1) each classifier is based on the same method (e.g. naive Bayes) used in the non-hierarchical approach, (2) the probabilities of terms are estimated using a maximum likelihood estimator, and (3) each document is represented using a constant length feature vector employed uniformly at each level in the hierarchy.

The work on topic spotting by Wiener *et al.* [130] (which was explained in detail in section 3.8) inspired us to try our approach using a hierarchical mixture of experts (HME) model that will be explained in detail in the following chapters. During the development of this dissertation they published a sequel of their work applied to hierarchical classifiers [129]. Their model uses a meta-topic network that could be equivalent to the gating network in the HME model. They have published results using a two level hierarchy on the Reuters collection. Their results are competitive with other methods.

3.11 Summary

This chapter has presented a detailed survey of the different methods that have been applied in text categorization. We can generalize that all these methods are based on supervised learning techniques, most of them taken from machine learning. Most of these approaches have explored the problem of text categorization without taking into account the hierarchical structure of the classification vocabulary. The few attempts in using this hierarchical structure have unfortunately concentrated on the Reuters collection which does not have a well defined hierarchical structure. On the other hand OHSUMED, which has a nice hierarchical structure based on the Medical Subject Headings (MeSH), has been explored by very few researchers possibly due to both the size of the collection, as well as the large number of categories (more than $14,000$) and scalability limitations of many of the proposed techniques. Moreover, none of the previous studies have used the hierarchy of MeSH categories, a classification scheme that has a rich knowledge structure. This leads us to conclude that the exploration of hierarchical methods for text categorization is an open issue that can be addressed in this dissertation.

Text categorization is an area of research that has captured the attention of many researchers because it is a challenging high dimensional problem that calls for effective techniques capable of efficiently handling a high volume of data. It is still a growing research area. Only recently a survey on text categorization was published by Sebastiani [120][2].

The next chapter will present a detailed description of our approach to text categorization, emphasizing the use of the hierarchical structure of the classification vocabulary, and experimental results using this method and different text categorization algorithms.

[2]We thank Dr. Fabrizio Sebastiani for making available his compiled bibliography[120].

CHAPTER 4
HIERARCHICAL MIXTURES OF EXPERTS MODEL

The goal of this dissertation is to build a classifier that exploits the hierarchical structure of the classification scheme. As we mentioned before, a few researchers have explored hierarchical text categorization [63, 84, 90, 96, 129]. Most of these methods use the high level nodes as filters that control the activation of low level nodes. Weiner et.al. [130] proposed a method in which the high level filter (which is represented by a multi-layer perceptron) computes the probability that a group of topics is present in a document. Although their research involves a shallow classification hierarchy, their approach uses a mixture model where the high level filter learns the best way of combining the lower level categories to obtain the best categorization result. This work inspired us to explore the use of the *mixtures of experts* (ME) model for text categorization. In particular we are interested in using a hierarchical version of the ME model. This chapter presents the theoretical basis of the *Mixtures of Experts* model, and their natural extension the *Hierarchical Mixtures of Experts* (HME). We present the connection between the HME and the hierarchical indexing structure, concluding with the presentation of our adaptation of the HME model for text categorization that will be used in this thesis to build the hierarchical classifiers.

4.1 Mixtures of Experts Model

The *mixtures of experts* (ME) model is based on the “divide and conquer” principle in which a large problem is divided into many smaller, easier to solve problems whose solutions can be combined to yield a solution to the complex problem. The mixtures of experts model consists of a set of *experts*, which model conditional probability processes, and a *gate* which combines the probability of the experts (see Figure 4.1).

The ME model is based on the classical decomposition of target under the assumption that there is an intermediate process Z that relates the inputs X with the outputs Y. This decomposition can be expressed in probabilistic terms as follows:

$$P(Y|X) = \sum_{Z} P(Z|X)P(Y|X,Z) \tag{4.1}$$

where Z is a set of hidden or missing data, and the sum is over all configurations of Z. The hidden data indicates which expert is responsible for generating each point. In a training set of N points, Z is made up of vectors $\mathbf{z}^{(n)}$, for $n = 1, \ldots N$, that have binary values of 0 or 1. When an expert ε_i ($i = 1, \ldots, I$ where I is the number of experts) is responsible for generating the example, the corresponding component $z_i^{(n)}$ is set to 1 otherwise it is set to 0.

$$z_i^{(n)} = \begin{cases} 1 & \text{if expert } \varepsilon_i \text{ generated } \mathbf{y}^{(n)} \text{ from } \mathbf{x}^{(n)}; \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

This decomposition implies that each pair $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$ is generated by a single specific process, indicated by the state of the vector $\mathbf{z}^{(n)}$, generating the output $\mathbf{y}^{(n)}$ given the input $\mathbf{x}^{(n)}$. The modeling of this decomposition needs two tasks to be addressed: 1) to model the processes Z, given X; and 2) to model the assignment of the output Y, given X and specific process Z. In the ME model these tasks are performed by two basic components: a gate node and two or more experts nodes. The gate models the probabilistic assignments of inputs to processes, $P(\mathbf{z}^{(n)}|\mathbf{x}^{(n)}, \mathbf{v})$ given parameter $\mathbf{v}$. For each process i an expert node models the probabilistic generation of outputs $P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, z_i^{(n)} = 1, \mathbf{w_i})$ given the inputs and the parameter $\mathbf{w_i}$.

In a general mixtures of experts architecture there are I experts $\{\varepsilon_i\}_1^I$ and a gate G. Together they model the conditional density of targets $Y = \{\mathbf{y}^{(n)}\}_1^N$ given inputs $X = \{\mathbf{x}^{(n)}\}_1^N$ as:

$$P(Y|X,\theta) = \prod_{n=1}^{N} \sum_{i=1}^{I} P(z_i^{(n)} = 1|\mathbf{x}^{(n)}, \mathbf{v}) P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, z_i^{(n)} = 1, \mathbf{w}_i) \tag{4.3}$$

where θ is the overall set of parameters $\theta = \{\mathbf{v}, \{\mathbf{w_i}\}_1^I\}$. To improve readability of formulas from here on we will use the following shorthand notation proposed by Waterhouse [128]:

$$P(\varepsilon_i|\mathbf{x}^{(n)}, \mathbf{v}) \equiv P(z_i^{(n)} = 1|\mathbf{x}^{(n)}, \mathbf{v}) \tag{4.4}$$

$$P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \varepsilon_i, \mathbf{w_i}) \equiv P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, z_i^{(n)} = 1, \mathbf{w_i}) \tag{4.5}$$

The expected value of the probability density function of equation 4.3 is used for predicting the values of target $\mathbf{y}^{(n)}$ given $\mathbf{x}^{(n)}$ according to the following model:

$$\begin{aligned} \hat{\mathbf{y}}^{(n)} = E(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \theta) &= \sum_{i=1}^{I} P(\varepsilon_i|\mathbf{x}^{(n)}, \mathbf{v}) E(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \varepsilon_i, \mathbf{w}_i), \qquad (4.6) \\ &= \sum_{i=1}^{I} g_i^{(n)} \hat{\mathbf{y}}_i^{(n)} \end{aligned}$$

where the short-hand notation $g_i^{(n)} = P(\varepsilon_i|\mathbf{x}^{(n)}, \mathbf{v})$ and $\hat{\mathbf{y}}_i^{(n)} = E(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \varepsilon_i, \mathbf{w}_i)$.

In other words, given an input $\mathbf{x}^{(n)}$, each expert ε_i makes a prediction $\hat{\mathbf{y}}_i^{(n)}$ of the target $\mathbf{y}^{(n)}$. The gate combines these predictions with its outputs $\{g_i^{(n)}\}_1^I$ to give an overall prediction of the model (4.6). Figure 4.1 shows an schematic representation of a ME model. The gate outputs $\{g_i^{(n)}\}_1^I$ can be interpreted as estimates of the probabilities of selecting each of the experts I given the input $\mathbf{x}^{(n)}$. An appropriate parameterization of the gate is a probabilistic classifier, e.g., a logistic regression model:

$$P(\varepsilon_i|\mathbf{x}^{(n)}, \mathbf{v}) = \frac{1}{1 + \exp(-(\mathbf{v}^T \mathbf{x}^{(n)}))} \tag{4.7}$$

in which the gate consist of a $d + 1$ dimensional parameter vector $\mathbf{v}$.

The form of the experts is chosen in such a way that it fits the particular problem at hand, for example a linear regression model, multi-layered perceptrons or

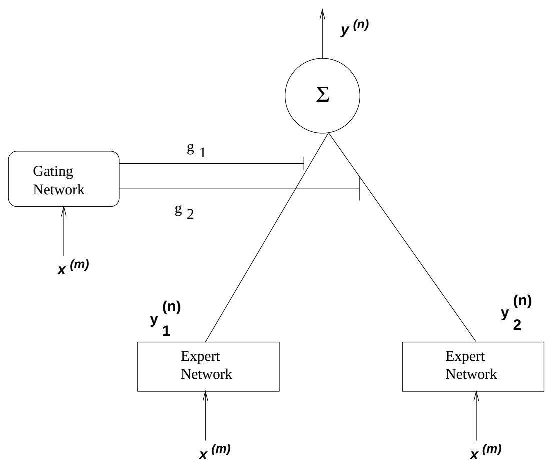

Figure 4.1: Example of a Mixtures of Experts (ME) model with two experts, ε_1 and ε_2 and a gate G

radial basis functions. The experts can take a form such that the expected value of their probability density is consistent with the form of the problem, although it is desirable to use experts which have simple form since their optimization is easier and they may be easily interpreted.

Waterhouse [128] gives a nice interpretation of the mixtures of experts model using belief networks. A belief network, also called Bayesian network, is a graphical representation of the joint probability statements. These graphical models are appealing because they allow us to represent different models within a unified scheme. Bayesian networks are an active focus of current research, and a variety of algorithms have been proposed for learning and using them for inference [88]. The believe network from Figure 4.2 expresses the assumption that the target Y is dependent on the input X and the multinomial random variable Z.

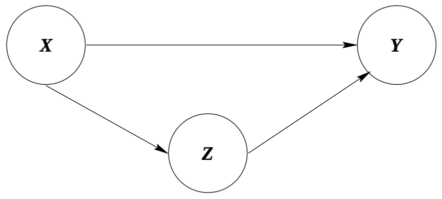

Figure 4.2: Belief Network for the Mixtures of Experts model

4.2 Hierarchical Mixtures of Experts

The *Hierarchical Mixtures of Experts*(HME) model is a supervised feedforward network that may be used for classification or regression [60]. As the ME model, it is based on the principle of "divide and conquer" and can be considered a generalization of the ME model in which the children nodes can also be mixtures models. Figure 4.3 shows an example of an HME model with two levels and binary branching nodes. An intuitive interpretation of the HME model is that each process is itself composed of a decomposition into processes that are selected stochastically.

Various methods have been proposed for decomposing large problems using the "divide and conquer" approach. The simplest approach is to divide the problem into sub-problems that have no common elements, also called a "hard split" of the data. The optimum solution of the smaller problems can then be chosen on a "winner-takes-all" basis. Classification and Regression Trees (CART) [8] are based on this principle. Stacked Generalization [132] also uses a hard split of the data and a weighted sum with weights derived from the performance of the smaller problems in their partition space. In contrast, HME divides the large problem into sub-problems that can have

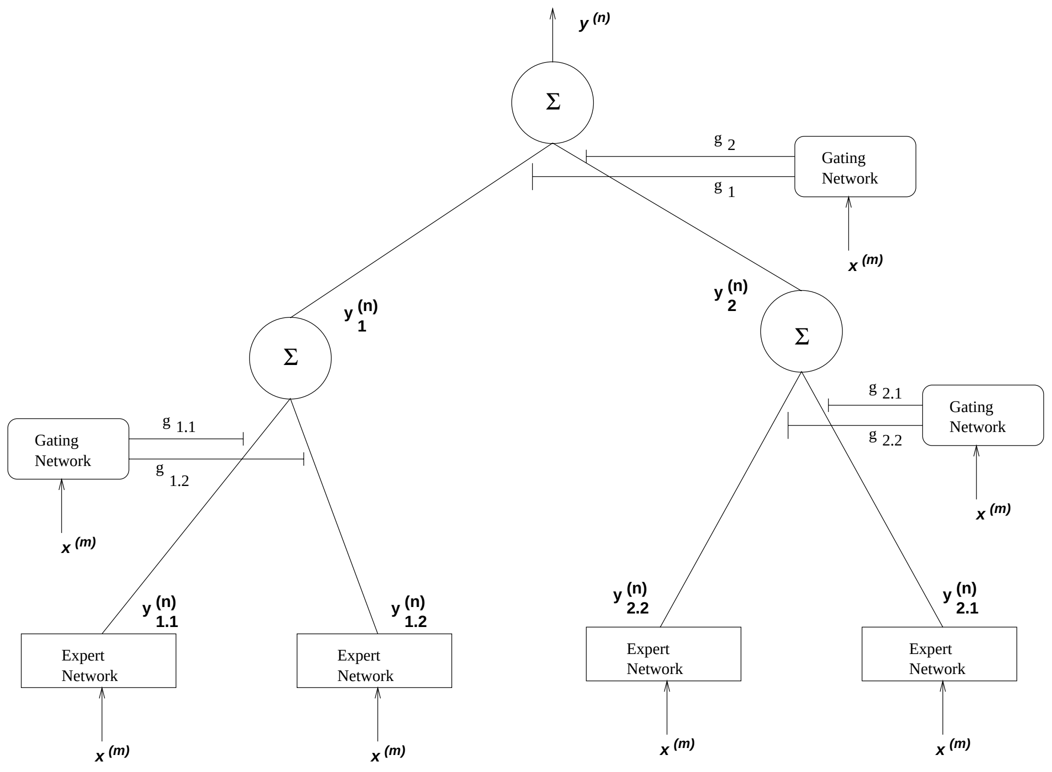

Figure 4.3: Hierarchical Mixtures of Experts model

common elements – a “soft split” of the elements into a series of overlapping clusters. The outputs of the simple problems are combined stochastically to obtain a global solution.

The HME presented in Figure 4.3 may be seen as a cascade of networks that works in a “bottom-up” fashion: the input is presented to the experts that generate an output, then the output of the experts are combined by the second level gates, generating a new output. Finally the outputs of the second level gates are combined by the root gate to produce the appropriate result $\mathbf{y^{(n)}}$.

To obtain the conditional probability density of the HME model shown in

Figure 4.3 we define an indicator variable $z_i^{(n)}$ for each intermediate node in the tree. $z_i^{(n)}$ is 1 if the node i is in the path from the root node to the terminal node that was considered to generate the data at time n, and 0 otherwise. We use individual gates G_j with parameter $\mathbf{v}_j$ to model each conditional probability at the intermediate nodes and an expert ε_i with parameter $\mathbf{w}_i$ to represent each terminal node. Given independent identically distributed data, we can write the conditional probability of the HME model with two levels (shown in Figure 4.3) as:

$$\begin{aligned} P(\mathbf{Y}|\mathbf{X},\theta) &= \prod_{n=1}^{N}\sum_i P(z_i^{(n)}=1|\mathbf{x}^{(n)},\mathbf{v}_0) \\ &\quad \sum_j P(z_j^{(n)}=1|z_i^{(n)}=1,\mathbf{x}^{(n)},\mathbf{v}_i)P(y^{(n)}|\mathbf{x}^{(n)},z_i^{(n)}=1,z_j^{(n)}=1,\mathbf{w}_j) \end{aligned} \tag{4.8}$$

where the indices i and j represent the first, and second levels respectively. Using a shorthand notation similar to the one presented for the ME model we write the prediction of the expert ε_j as $\hat{y}_j^{(n)}$ and th jth prediction of a gate as $g_j^{(n)}$. The expected value of $y^{(n)}$ is:

$$\begin{aligned} \hat{y}^{(n)} &= E(y^{(n)}|x^{(n)},\theta) \\ &= \sum_i P(z_i^{(n)}=1|\mathbf{x}^{(n)},\mathbf{v}_0) \\ &\quad \sum_j P(z_j^{(n)}=1|z_i^{(n)}=1,\mathbf{x}^{(n)},\mathbf{v}_i)E(y^{(n)}|\mathbf{x}^{(n)},z_i^{(n)}=1,z_j^{(n)}=1,\mathbf{w}_j) \\ &= \sum_i g_i^{(n)}\sum_j g_j^{(n)}y_j^{(n)} \end{aligned} \tag{4.9}$$

In the original HME model proposed by Jordan and Jacobs all the networks in the tree are linear (perceptrons). The gating networks are also generalized linear functions. The i^{th} output of the top level gating network is the “softmax” function of the intermediate variable ψ_i [9, 86]:

$$g_i = \frac{e^{\psi_i}}{\sum_{j=1,\ldots,I} e^{\psi_j}} \tag{4.10}$$

where I is the number of child nodes of the gating network, and the intermediate variable ψ_i is defined as:

$$\psi_i = \mathbf{v}_i^T \mathbf{x} \tag{4.11}$$

where ^{T} is the transpose operation. The g_is are positive and sum to one for each $\mathbf{x}$. They can be interpreted as providing a “soft” partitioning of the input space.

Similarly, the gating networks at the lower levels are also generalized linear systems. The output of the j^{th} unit in the i^{th} gating network at the second level of the architecture, denoted by $g_{j|i}$, is defined as:

$$g_{j|i} = \frac{e^{\psi_{ij}}}{\sum_{j=1,\ldots k} e^{\psi_{ij}}} \tag{4.12}$$

where k is the number of child nodes of the gating network, and the intermediate variable ψ_{ij} is defined as follows:

$$\psi_{ij} = \mathbf{v}_{ij}^T \mathbf{x} \tag{4.13}$$

Note that since both the g's and the y's depend on the input $\mathbf{x}$, the output is a nonlinear function of the input.

4.3 HME Model for Text Categorization

We build a variation of the HME model that is adapted to text categorization. In general, the classification task can be a 1-of-K (multinomial classification) task or a k-of-K (multi-way classification) task. 1-of-K classification tasks can be viewed as a competition problem. Text categorization is a k-of-K task since a document could be assigned to one or more categories simultaneously. k-of-K classifications are equivalent to K independent 1-of-2 classifications [86, 107]. In our model, a gate has a single output that behaves like a binary switch. It will have value 1 if any of the categories of its descendants have been assigned to the training document.

This can be interpreted in terms of a hierarchical classification as the fact that a general concept is present in the document, e.g., the concept “Heart diseases” is present in a document that talks about the specific concept “Coronary Thrombosis”. Figure 4.4 shows an example of our hierarchical classifier.

Our categorization task starts at the root node and the gate decides whether the most general concept is present in the document. If this is true, then all the second level nodes are activated and the process repeats again until it reaches the leaf nodes. Observe that only the experts connected to gates that have value 1 are

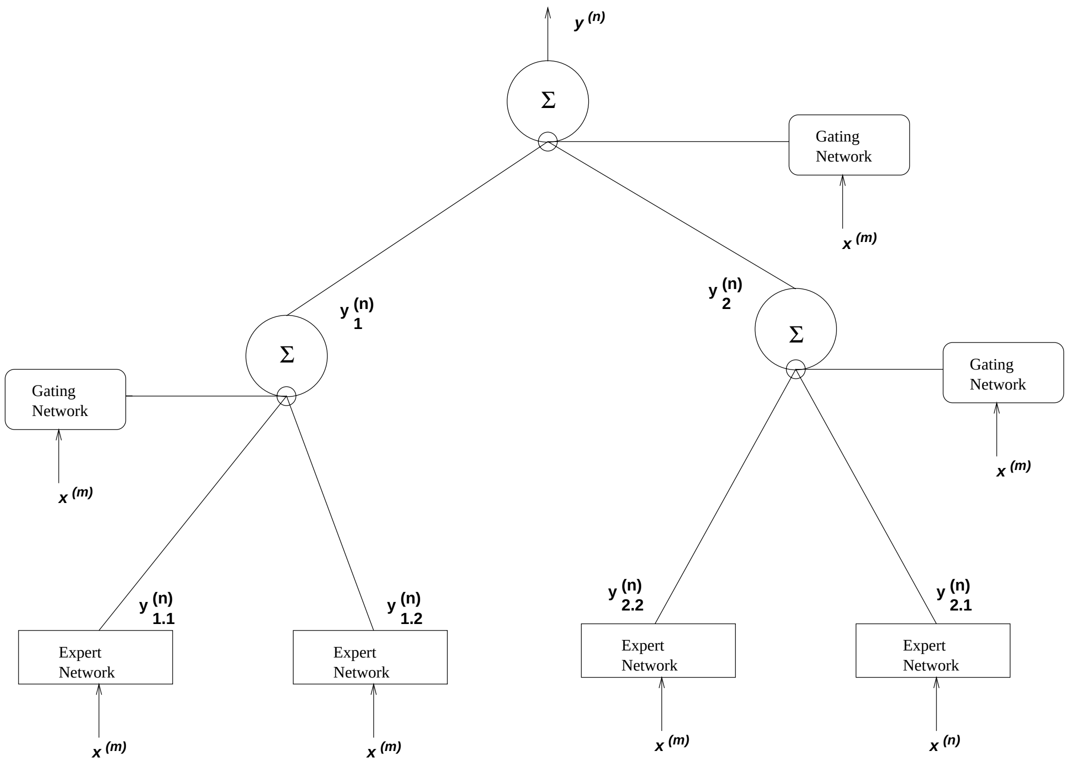

Figure 4.4: Modified Hierarchical Mixtures of Experts model

activated. This is equivalent to have a HME model in which the output of the gates is a threshold function. Observe that since gates and experts depend only on the input $\mathbf{x}$ we can compute their output in a "bottom up" or "top down" fashion, however training the HME is usually done in a top down fashion. Since our model takes a binary decision on the upper level nodes, proceeding in a "top down" fashion would reduce the number of nodes activated reducing the response time for classification.

To give a statistical interpretation of our model we define $Z_0, \ldots, Z_j$ as the path of gates from the root node to the gate j that is the parent of an expert ε_k that assigns category k. Let $\mathbf{x}^{(n)}$ be the input features that represent a document, and

$\mathbf{y}^{(n)}$ the output vector of the categories assigned to the document. The probabilistic interpretation of our hierarchical model is as follows:

$$\begin{aligned}\hat{\mathbf{y}}^{(\mathbf{n})} &= E(P(\mathbf{y}^{(\mathbf{n})}|\mathbf{x}^{(\mathbf{n})},\theta)) \\ &= \sum_{k=1}^{I} P(Z_0=1|\mathbf{x}^{(n)},\mathbf{v_0})P(Z_1=1|Z_0=1,\mathbf{x}^{(n)},\mathbf{v_1})\dots \\ &\quad P(Z_j=1|Z_0=1,\dots,Z_{j-1}=1,\mathbf{x}^{(n)},\mathbf{v_j})E(\mathbf{y}_k^{(n)}|\mathbf{x}^{(n)},Z_0=1,\dots,Z_j=1,\mathbf{w_k})\end{aligned} \tag{4.14}$$

The hierarchical structure in a HME model is predefined[1], and generally defined in terms of the number of classes that are present in the training set. In the case of a classifier for text categorization we will use the hierarchy of the classification vocabulary. This allows us to integrate information about the relationships between the different categories of the domain which in flat classifiers are ignored.

There are several alternatives for training a HME model. Jordan and Jacobs [60] and Waterhouse [128] use a method based on expectation maximization. They assume that the classification follows a multinomial model, which implies that an object can be assigned to one and only one of the multiple categories available for classification. To allow for multi-way classification we will use backpropagation neural networks in both gates and experts and use a gradient descent method for training. The gates are trained to recognize whether or not any of the categories of its descen-

[1]We must note that Waterhouse [128] has proposed a HME model that dynamically generates the hierarchical structure.

dants is present in the document. The experts are trained to recognize the presence or absence of particular categories.

The backpropagation networks that we use have three layers. We have tested several configurations and these results will be discussed in detail later. Figure 4.5 shows an example of the neural networks used in this work. In general, our neural networks have m nodes in the input layer corresponding to the set of m features selected for each expert (or gate), the middle layer has n nodes, and the output layer is a single node.

Given an appropriate set of features and a training set of manually categorized documents, the backpropagation network learns to assign the category (or concept in the case of gates). Observe that for experts and gates the set of positive examples is different. The set of positive examples for the experts is a subset of the positive examples of any ancestor gate. As a consequence, two identical neural networks trained with these different subsets learn different probabilistic functions. The backpropagation neural network as an expert node learns to use the input to estimate the desired output value (category), while as a gate it computes the confidence value of the combined outputs of its children.

This chapter presented the ME and HME models that form the foundation of our hierarchical text categorization approach. In the next chapter we describe the application of our approach to the particular categorization problem selected for this dissertation.

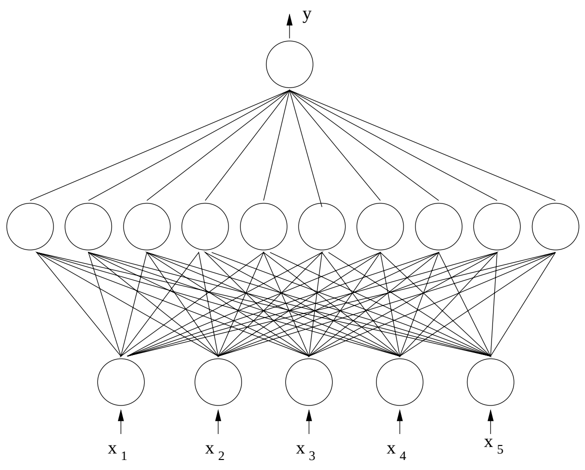

Figure 4.5: Example of a backpropagation network with 5 input nodes

CHAPTER 5
IMPLEMENTATION OF THE HME MODEL FOR TEXT CATEGORIZATION

This chapter describes the implementation of our hierarchical classifier in the context of the Unified Medical Language System (UMLS) Metathesaurus [97] and the OHSUMED collection. The classifier is an implementation of the HME model configured with backpropagation neural networks at each expert and gate node as described in the previous chapter.

We are interested in the UMLS Metathesaurus because of its hierarchical inter-concept links. The UMLS is a vocabulary system created by combining 79 vocabularies from the health science[1]. The metathesaurus is one major part of the UMLS. It consists of about $350,000$ concepts with various types of inter concept relationships represented. In this study we limit ourselves to MeSH (Medical Subject Headings)[2] subset of UMLS which is one of the 79 component vocabularies. This is because documents in the OHSUMED test collection are a subset of MEDLINE, and thus have been manually categorized with MeSH terms. Each MEDLINE document is assigned between 8 and 10 MeSH concepts by indexers at the National Library of Medicine.

[1]We use the 1999 version.

[2]Observe that we could have used the MeSH hierarchy directly. We decided to use the UMLS hierarchy because it provides a conceptual mapping that has been extended to many areas of the health sciences.

Thus our categorization task is a multi-way classification problem with gold standard being manual indexing.

Interestingly, the manual assignment of a high level MeSH category is not automatically determined by the assignment of its lower level categories. That is, the fact that a document is assigned the category “angina unstable” does not automatically grant it the assignment of any of the ancestors in the tree (“Heart diseases”, “Myocardial Ischemia”, “Coronary Diseases”, or “Angina Pectoris”). In fact the manual assignment of such high level categories is usually done when the MEDLINE document is about the topic at the associated level of generality or abstraction. Therefore in our model, each nonterminal node is represented by two networks. The first is the expert network for the node’s category while the second is a gating network representing the general concept at that level of the classification scheme. Thus at the “Heart Diseases” node there is a gate that learns to recognize the general *concept* (representing all the documents that are about any of its descendants), and an expert network that learns to assign this specific category “Heart Diseases”. Note that from this point on unless explicitly specified, we mean both categories and concepts when we use the word ‘category’.

For the purpose of comparing results with other studies we will show the results obtained using only the MeSH subtree of “Heart Diseases” (Figure 5.1 shows a part of this hierarchy). However, our method especially given its top-down processing, is general and can be applied to the whole set or to any other subset of the UMLS.

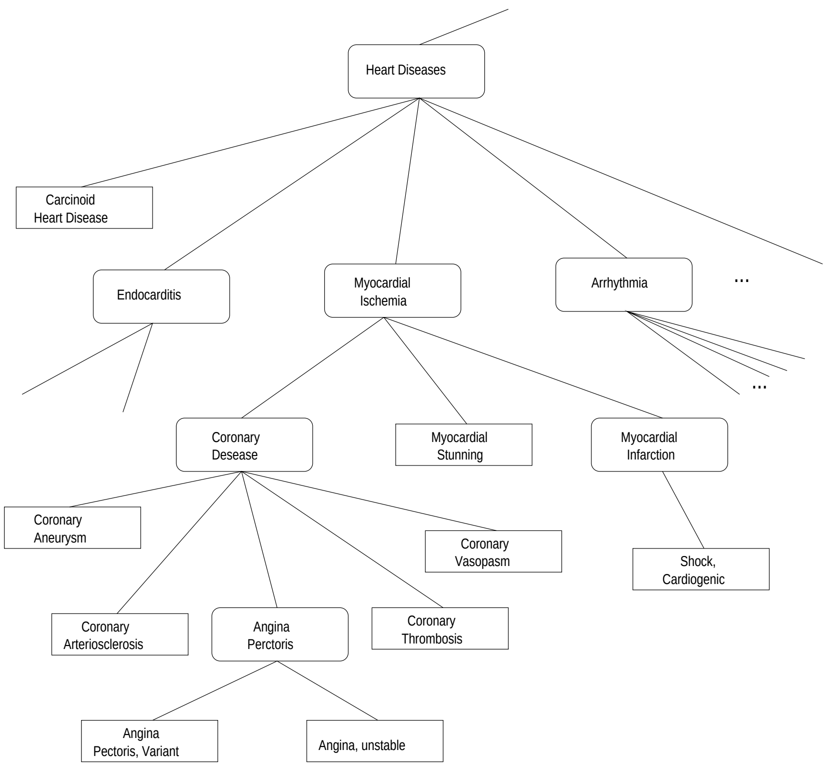

Figure 5.1: A part of the UMLS hierarchy for the heart diseases subtree

The "Heart Diseases" subset consists of 119 categories and is organized as a five level hierarchy. Appendix A shows the complete schema of the Heart Diseases subtree.

5.1 Feature Selection

In text categorization the set of possible input features consists of all the different words that appear in a collection of documents. This is usually a large set since even small text collections could have hundreds of thousands of features. Reduction of the set of features to train the neural networks is necessary because the performance of the network and the cost of classification are sensitive to the size and quality of the input features used to train the network [133]. A first step towards reducing the size of the feature set is the elimination of stop words, i.e., words that do not carry meaning by themselves such as articles and prepositions [115], and the use of stemming algorithms [33, 115]. Even after that is done the set of features is typically too large to be useful for training a neural network.

Two broad approaches for feature selection have been presented in the literature: the wrapper approach, and the filter approach [59]. The wrapper approach attempts to identify the best feature subset to use with a particular algorithm. For example, for a neural network the wrapper approach selects an initial subset and measures the performance of the network; then it generates an "improved set of features" and measures the performance of the network. This process is repeated until it reaches a termination condition (either a minimal value of performance or a number of iterations). The filter approach, which is more commonly used in text categorization, attempts to assess the merits of the feature set from the data alone. The filtering approach selects a set of features using a preprocessing step, based on the training data. In this dissertation we use the filter approach applying three methods that have been reported in previous works: correlation coefficient, mutual information, and odds ratio. During feature selection we first delete all instances of 571 stop words from the MEDLINE records (this list is shown in appendix B), and then use a variation of Lovins' algorithm to stem the remaining words [33, 79, 115]. Lovins' stemming algorithm is an iterative longest match stemmer which consists of a set of rules, which are iteratively applied to reduce a word to its stem, and a set of conditions under which stemming cannot be applied (called exception rules). The algorithm finds the longest suffix that satisfy the set of restrictions specified by the exception rules. The longest valid suffix is removed from the word to obtain its stem which is converted to a standard form (e.g. "believes" is stemmed to "belief"). We eliminate those stems that occur in less than 5 documents in the training collection. Since feature selection is done for each category, based on its zone (explained later) we also remove stems that occur in less than 5% of the positive example documents. We then rank the remaining stems by the feature selection measure and select a pre-defined number of top ranked stems as the feature set.

5.1.1 Correlation Coefficient

Correlation coefficient C is a feature selection measure proposed by Ng *et al.* [96] and is defined as:

$$C(w,c) = \frac{(N_{r+}N_{n-} - N_{r-}N_{n+})\sqrt{N}}{\sqrt{(N_{r+}+N_{r-})(N_{n+}+N_{n-})(N_{r+}+N_{n+})(N_{r-}+N_{n-})}} \tag{5.1}$$

where $N_{r+}(N_{r-})$ is the number of positive examples of category c in which feature w occurs(does not occur), and $N_{n+}(N_{n-})$ is the number of negative examples of category c in which feature w occurs(does not occur). This measure is derived from the χ^2 measure presented by Schütze *et al.* [119], where $C^2 = \chi^2$. The correlation coefficient can be interpreted as a "one-side" χ^2 measurement. The χ^2 measure has been reported as a good measure for text categorization by Yang and Petersen [142]. The correlation coefficient promotes features that have high frequency in the relevant examples but are rare in the non relevant documents. When features are ranked by this method, the positive values correspond to features that indicate presence of the category while the negative values indicate absence of the category. In contrast, the χ^2 ranks features higher if they more strongly indicate the presence or the absence of a category. That is, more ambiguous features are ranked lower. We compared the χ^2 and correlation coefficient for feature selection using neural networks with the same architecture. We found that the neural networks trained with features selected using correlation coefficient outperformed those trained using χ^2 in 78 out of 103 categories. This confirms similar results reported by Ng et al. [96]. Note that Yang and Pedersen [142] use an average of the χ^2 value across categories to measure the goodness of a term in a global sense, while we use it for local (category-level) feature selection.

In contrast with mutual information, both χ^2 and correlation coefficient produce normalized values because they are based on the χ^2 statistic. However, the normalization does not hold for low populated cells in the contingency table. This makes the scores of χ^2 and correlation coefficient for low frequency terms unreliable. This is one reason for removing rare features as described before.

5.1.2 Mutual Information

Mutual information is a measure that has been used in text categorization by several researchers [119, 142]. This method is based on the mutual information concept developed in information theory. For a feature w and a category c it is defined as:

$$I(w, c) = \log \frac{P(w \wedge c)}{P(w) \times P(c)} \tag{5.2}$$

where $P(w)$ is the probability of the term w occurring in the whole collection, $P(c)$ is the probability of the category c occurring in the whole collection, and $P(w \wedge c)$ is their joint probability.

Yang and Pedersen [142] used mutual information[3] to measure the goodness of a term in a global feature selection approach by combining the category specific scores of a term in two ways:

$$I_{avg}(w) = \sum_{i=1}^{n} P(c_i) I(w, c_i) \tag{5.3}$$

$$I_{max}(w) = \max_{i=1}^{n} \{I(w, c_i)\} \tag{5.4}$$

where n is the number of categories.

In contrast, we use feature selection to evaluate the goodness of a term with respect to individual categories. In other words, we do not average the values of mutual information over multiple categories. This variation may be sufficient to produce the different results that we obtain (described later). Yang and Pedersen also point out that the score produced by mutual information is strongly influenced

[3]There is some confusion with the term "mutual information". For instance, it has been used by other researchers [83] to refer to the measure that Yang and Pedersen [142] present as "information gain".

by the marginal probabilities of terms. This is evident from the following equivalent formula:

$$I(w, c) = \log P(w|c) - \log P(w) \tag{5.5}$$

For terms with equal conditional probability $P(w|c)$, rare terms will have higher scores than common terms. This implies that the scores of terms with extremely different frequencies might still not be comparable. Our frequency threshold described earlier, compensates for this effect.

5.1.3 Odds Ratio

Odds ratio was proposed originally by van Rijsbergen *et al.* [126] for selecting terms for relevance feedback. Odds ratio is used for the binary-valued class problem where the goal is to make a good prediction for one of the class values [127]. It is based on the idea that the distribution of features on the relevant documents is different from the distribution of features on the non-relevant documents. It has been recently used by Mladenić [90] for selecting terms in text categorization. The odds ratio of a feature w, given the set of positive examples c and negative examples $\bar{c}$ for a category c, is defined as follows:

$$OddsRatio(w, c) = \log \frac{P(w|c)(1 - P(w|\bar{c}))}{(1 - P(w|c))P(w|\bar{c})} \tag{5.6}$$

Observe that this formula can also be interpreted as the sum of the logarithm

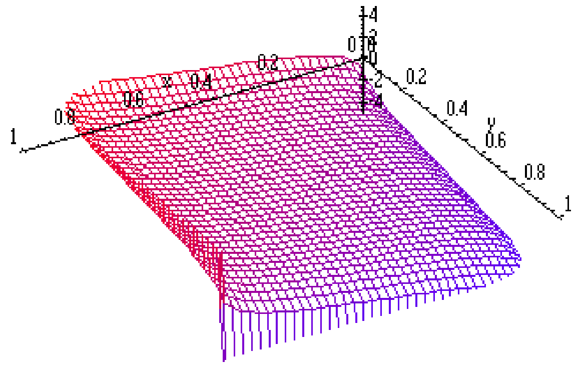

Figure 5.2: Graph of Odds Ratio

of the ratios of the distribution of the feature on the relevant documents ($\log \frac{P(w|c)}{(1-P(w|c))}$) and on the non-relevant documents ($\log \frac{(1-P(w|\bar{c}))}{P(w|\bar{c})}$). If a document appears in more than half of the relevant documents the logarithm of the ratio on the relevant documents is positive. In contrast a feature is penalized if it appears in more than half of the non-relevant documents. In other words, a feature that appears frequently in the relevant documents and infrequently in the non relevant documents will have a high score. Figure 5.2 shows a graph of the odds ratio[4]. The function presents singularity points when $P(w|c) = 1$ or when $P(w|\bar{c}) = 0$ (we map this case to the highest positive value). Also the logarithm is not defined when $P(w|c) = 0$ or when $P(w|\bar{c}) = 1$ (we map this case to the smallest negative value).

Mladenić [90] report that odds ratio was the most successful feature selection method for a hierarchical Bayesian classifier compared to mutual information, cross

[4]The x axis represents $P(w|c)$, while the y axis represents the $P(w|\bar{c})$

entropy, information gain, and weight of evidence.

In this study we select features for both expert and gating networks using correlation coefficient, mutual information and odds ratio methods.

5.2 Training Set Selection

A supervised learning algorithm requires the use of a training set in which each element has already been correctly categorized. One would expect that the availability of a large training set (such as OHSUMED) will be beneficial for training the algorithm. In practice this does not seem to be the case. The problem occurs when there is also a large collection of categories with each assigned to a relatively small number of documents. This then creates a situation in which each category has a small number of positive examples and an overwhelming number of negative examples. When a machine learning algorithm is trained to learn the assignment function with such an unbalanced training set, the algorithm will learn that the best decision is to not assign the category. The overwhelming amount of negative examples hides the assignment function. To overcome this problem an appropriate set of training examples must be selected. We call this training subset the "category zone". This notion of category zone is similar to the local regions described in Wiener *et al.* [130] , and Ng *et al.* [96] but is inspired by the query zone proposed by Singhal *et al.* [123] for text routing. Their "query zoning" is based on the observation that in a large collection a query will have a set of documents that constitutes its domain. Non-relevant documents that are outside the domain are easy to identify, but it is more difficult to differentiate between relevant and non-relevant documents within the query domain. Singhal *et al.* [123] define a procedure that tries to approximate the domain of the query and then they use this domain to train their routing method. We suggest that in text categorization, each category also has its own domain. It will be easier to train a learning algorithm with those documents from the category domain and also potentially achieve better categorization performance. We explore two different methods for building the category zone. The first method creates the category zone using a method similar to that presented by Singhal *et al.* [123]. This first category zone that we call centroid-based is created as follows:

1. Take all the positive examples for a category and obtain their centroid.
2. Using this centroid as a query perform retrieval and obtain the top $10,000$ documents. This subset will contain most if not all of the positive examples and many negative examples that are at least "closely related" to the domain of the category.
3. Obtain the category zone by adding any unretrieved positive examples to the set obtained in the previous step.

This method creates category zones that have at least $10,000$ documents and the size increases for categories that have positive examples outside the retrieved set.

The second method for creating the category zone uses a Knn approach in which the category zone consists of the set of K nearest neighbors for each positive example of the category. This method takes a more localized view of the zone, localized around each positive example. It also produces variable sized category zones. We explore several values of K (10, 50, 100 and 200). Our main concern with this method was to obtain a training set large enough to train a neural network without overfitting.

In summary, the neural networks at the nodes of our classifiers correspond to a natural hierarchical arrangement of our classification scheme – the "Heart Diseases" MeSH subtree of the UMLS Metathesaurus. Moreover, in order to parallel the human indexing performance in MEDLINE documents, the non-leaf nodes of our hierarchical classifier have both a gating network and an expert network. This chapter also describes the feature selection methods explored and zoning techniques to select training examples. In the next chapter we present a series of experiments that show the performance of the described methods in text categorization using the standard OHSUMED collection.

CHAPTER 6
ASSESSING THE NEURAL NETWORK BASED HME MODEL

Consistent with our research goal we presented in the previous chapter a classifier that is potentially capable of exploiting the hierarchical structure underling a classification scheme. In this chapter we address two specific questions that together explore the value of hierarchical text categorization: (1) Does our hierarchical classifier built on the HME model improve performance when compared to a flat classifier? (2) How does our hierarchical method compare with other text categorization approaches? We also address other aspects related to our model such as the effect of feature selection and training subset selection. With these research questions in mind this chapter presents in detail a series of experiments. We begin by describing in detail the OHSUMED collection and then describe the implementation of the classifiers tested. The chapter concludes with a presentation of results.

6.1 Experimental Collection

We use the OHSUMED collection [46], a subset of MEDLINE that we introduced in section 2.4.2. Specifically we use the split proposed by Lewis *et al.* [78]. Each record from this collection has several fields (see Figure 2.1). We use the following: title (.T), and abstract (.W). In the training set we also use the MeSH (.M) field which represents the manual categorization decisions for the MEDLINE documents. There are 233,455 records in OHSUMED that have titles, abstracts and MeSH categories (the remaining do not have abstracts). The first four years of data dated 1987 through 1990 (183,229 records) are used for training, and the year 1991 (50,216 records) is used for testing. We also use the 119 categories from the Heart Disease subtree of the Cardiovascular Diseases tree structure of the UMLS[1]. Observe that we could have used the MeSH hierarchy directly which is a subset of the UMLS hierarchy. These are identical within the "Heart Diseases" subtree. We choose to use the UMLS hierarchy because for future research it will allow us to build hierarchical classifiers for areas that have limited coverage in MeSH. Of the 119 categories of the Heart Diseases subtree only 103 categories have positive examples in the training set. Thus we limit our experiments to these 103 categories. We further divide this set of 103 categories into three sets:

- High frequency categories (HD-49): This includes all categories with at least 75 examples in the training set. This set contains 49 categories (which is the same as the set of high frequency categories used by Lewis *et al.*).

- Medium frequency categories (HD-28): This set includes all categories with frequencies between 15 and 74 in the training set. This set contains 28 categories (this is equivalent to the second set of categories used by Lewis *et al.*).

[1]As mentioned before we use the 1994 version of the UMLS. This avoids inconsistencies between the categories assigned to the OHSUMED documents and the current version of the UMLS.

- Low frequency categories (HD-26): This set includes all categories with frequencies between 1 and 14 in the training set. This set contains 26 categories.

We report results on these three subsets as well as on the complete set of categories (HD-119)[2]. The first two subsets allow us to analyze performance separately for different levels of positive evidence and also allow us to compare results with other published research with the same collection [78].

The 119 "Heart Diseases" categories form a 5 level tree where the first level corresponds to the root node and the fifth level has only leaf nodes. The number of gates in each level starting from the root is 1, 11, 9, and 3 (see Figure 6.1).

6.2 Baselines

Our first baseline represents a classical Rocchio classifier that is described in the next section. Our second baseline is a flat neural network classifier. Comparing the performance of the HME classifier against the flat classifier will allow us to answer our first research question. Comparing the HME method with a Rocchio classifier as well as with other published results will allow us to answer our second research question. Thus we have implemented a Rocchio classifier, a HME classifier, and a flat neural network classifier which are detailed next.

[2]We label this set HD-119 in order to stay consistent with the labeling in [135] even though there are actually only 103 categories with positive examples in the training set.

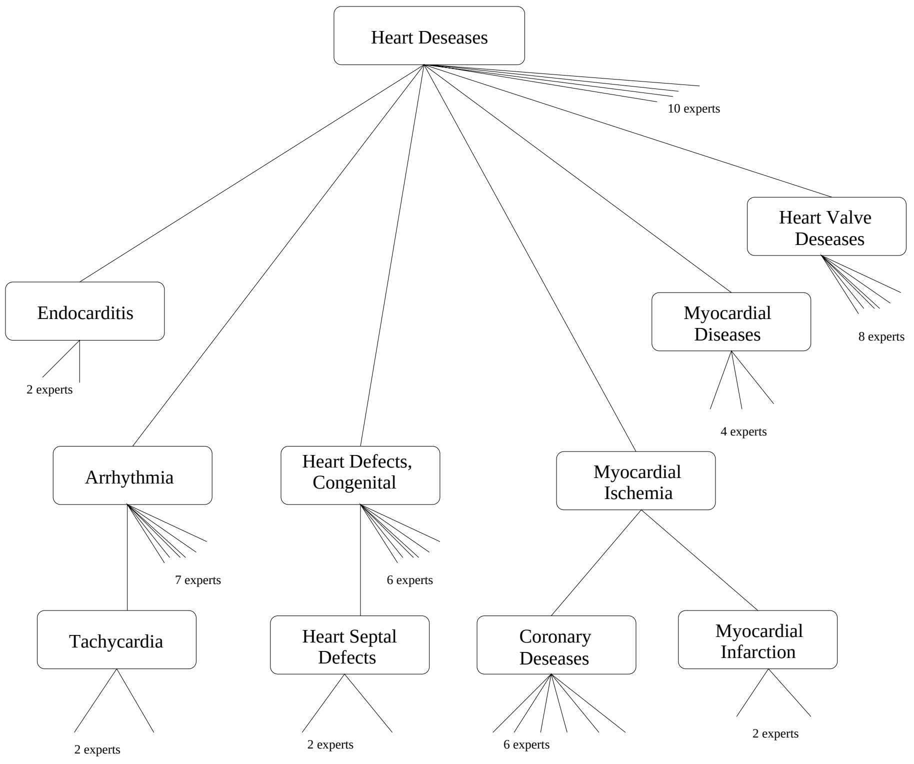

Figure 6.1: Tree for the 119 categories of the heart diseases sub-tree

6.3 Rocchio Classifier

We have implemented the Rocchio classifier described in section 3.1. The Rocchio classifier has been used by several researchers [65, 78, 135, 136]. As pointed out by Schapire *et al.* [118] most of these studies that use Rocchio as a baseline have constructed a weak version of the classifier. Schapire *et al.* also show that a properly optimized Rocchio algorithm could achieve quite competitive performance. Several techniques have been proposed to improve the effectiveness of Rocchio's method: better weighting schemes [122], query zoning [123], and dynamic feedback optimization [10]. We have noticed that Rocchio classifiers benefit from an optimal feature selection step. To make a fair comparison between the neural networks and the Rocchio classifiers we use the set of features selected using correlation coefficient and the same category zones used to train the neural network classifiers for each category. Observe that this is an important difference with respect to previously published research that use Rocchio classifiers. In all these studies the vector is computed over the whole set of features. Since we use feature selection measures that select features indicative of presence of the category, each classifier has its centroid vector defined in a different subspace (the sub-space of the selected features) generated from the category zone.

We build a Rocchio classifier by presenting training examples from the category zone and updating the weights of the classifier using Rocchio's formula shown in equation 3.2 (section 3.1). We then rank the full training collection according to the similarity with this classifier vector. A threshold (τ) on the similarity value that maximizes the F_1 measure (section 2.5) is selected. The *optimal* Rocchio classifier for a category is then a weighted vector of selected features along with the optimal similarity threshold.

During the evaluation phase we compute similarity between the optimal Rocchio classifier vector and the document we want to categorize. The similarity is the Euclidean distance between the vector that represents the classifier and the vector that represent the new document. The class is assigned if the similarity value is above the threshold τ.

6.4 Hierarchical Mixture of Experts

We built the HME classifier described in chapter 4 and represented in Figure 4.4. First a zone of domain documents is identified for each category as explained before in section 5.2. Next feature selection is applied within each category zone to extract the "best" set of features. We tested all three feature selection methods (described in 5.1) in our experiments. For each expert network a backpropagation neural network is trained using the corresponding category zone and the selected set of features. Similarly, each gating network is also a backpropagation network. However the training subset of a gate is the combined category zones of its descendants in the classification hierarchy. Feature selection for the gate is performed on this combined subset. This strategy of combining zones from descendant nodes for a gate is reasonable considering the fact that gates represent hierarchical concepts and not particular categories as described before in chapter 5.

The input feature vectors for documents are weighted using $tf \times idf$ weights where tf is the frequency of the term in the document, and idf is the inverse document frequency calculated as:

$$idf = log\frac{N}{nc} \tag{6.1}$$

where N is the number of documents in the training collection and nc is the number of documents that contain the term in the training collection.

Experts and gates are trained independently using the following parameters: learning rate = 0.5, error tolerance = 0.01, maximum number of epochs = $1,000$. These parameter values are fixed for all our experiments. The training of each network takes between 15 to 30 minutes for an expert (depending on the number of examples), and around 60 to 90 minutes for a gating network using a HP-700 workstation. Using 15 workstations and a dynamic scheduling program specifically designed for this task we trained the 103 experts and the 21 gating networks in about 8 hours.

Once experts and gates have been trained individually we assemble them according to the hierarchical structure of the UMLS “Heart Diseases” subtree. Since the output of each network is a real value between 0 and 1, we need to transform each output value into a binary decision. This step is called thresholding. We do this by selecting thresholds that optimize the F_1 values for the categories. We use the complete training set to select the optimal thresholds. Since we are working with a modular hierarchical structure we have several choices to perform thresholding. Our approach is to make a binary decision in each of the gates and then optimize the threshold on the experts using only those examples that reach the leaf nodes.

Observe that computing the optimal thresholds for binary decisions at the gates and the experts is a multidimensional optimization problem. We decided to optimize the gates by grouping them into levels and finding the value of the threshold at each level that maximizes the average F_1 value for all the experts. Each expert's threshold is then optimized to maximize the F_1 value of the examples in the training set that reach the expert. In order to constrain the potentially explosive combination of parameters we decide to fix the thresholds for the gates across all our experiments. For this purpose we conducted a preliminary experiment in which we sought the best combination of thresholds per level varying each threshold over fixed values and computing performance on the whole training set. The optimal thresholds were set to 0.01, 0.005, 0.01 and 0.01 for levels 1(root), 2, 3 and 4 respectively. These values were obtained by selecting the best results over $1,764$ threshold combinations (0.005, 0.01, 0.05, and 0.10 for level 1, 0.005, 0.01, 0.05, 0.10, 0.15, 0.20, ... 0.95 for levels 2, 3 and 4)[3]. This experiment was run using correlation coefficient for feature selection and a standard configuration of 25 input nodes and 50 hidden nodes. The best threshold values are fixed across all runs.

The test set is processed using the trained networks assembled hierarchically

[3]Since we only have three gates in level 4 we set their optimal values to the same values of the gates in level 3. This gives us $4 \times 21 \times 21 = 1,764$ possible combinations for the thresholds in the gates.

with the established thresholds for each level of gates and each expert network.

6.5 Flat Neural Network Classifier

In order to assess the advantage gained by exploiting the hierarchical structure of the classification scheme, we built a flat neural network classifier. We decided to build a flat modular classifier that is implemented as a set of 103 individual expert networks. This is similar to our Rocchio model where the training phase results in a set of 103 classifier vectors. In this model the experts are trained independently using the optimal feature set and the category zone for each individual category. The thresholding step is performed by optimizing the F_1 value of each expert using the entire training set. These are the values that we report in the next section for the flat neural network classifier.

6.6 Results

As stated before, we report results on the "Heart Diseases" sub-tree of the UMLS. We present results on this set of 103 categories (HD-119) and on three frequency-based subsets of categories HD-49, HD-28 and HD-26 as defined before in section 6.1.

6.6.1 Feature Selection and Neural Network Architecture

It may be observed that network architecture and feature selection methods must be studied in combination. In fact, at a basic level feature selection is one of the factors that define the configuration of the network. Given the many complex combinations in terms of feature selection methods and numbers of nodes in the different layers for both the expert and gating networks we approach the problem in stages. Specifically we follow a top-down approach that optimizes first the gates and then the experts.

While studying the gating networks we used experts with 25 input features and 50 nodes in the hidden layers. We then explored 5, 10, 25, 50, 100, and 150 input features for the gating networks with hidden layers that had twice the number of input nodes. We also tried all three feature selection methods (Mutual Information, Odds Ratio and Correlation Coefficient). This experiment was done only on the "high frequency categories" (HD-49) because they allow appropriate training of the neural networks and also because the variance between different training runs is smaller than the variance for lower frequency categories.

Interestingly the differences between the three feature selection methods on the gating networks are not significant. Thus we only report results on the 18 different combinations obtained using correlation coefficient in the gates[4]. Table 6.1 shows an increase in performance between 5 and 25 features and a slight decrease for networks with larger number of input nodes. It is possible for this slight decrease to be caused by the limit in the number of iterations $(1,000)$ that a network was allowed to run during training. Usually a larger network needs more iterations on the training set to converge to an optimal value. Observe that all the three feature selection methods

[4]Six different sizes for the input layer of the gates multiplied by 3 different feature selection methods for the expert networks.

show no significant differences for the expert networks as was also observed for the gates. This was somewhat surprising since Yang and Pedersen [141] reported that mutual information does not perform well compared to other methods. As noted before, instead of averaging values across multiple classes to find the merit of the features from a global perspective as in [141], we use mutual information for local feature selection. We also discard rare terms that will in general be ranked very high by mutual information.

We further explored feature selection using mutual information by running our experiments without discarding rare terms and selecting the top 25 features. The average precision for the HD-49 subset was 0.05 and 0.20 for the flat neural network and the HME model respectively. This is significantly lower than the performance obtained when we discard low frequency terms and shows conclusively that mutual information alone is not a good feature selection measure unless we address its major weakness and discard low frequency terms. This might also be addressed by selecting a larger number of input features. However, this will go against our goal of reducing the number of input features to improve training and processing time for the neural networks.

After optimizing the gates we turned our attention to the expert networks. We first addressed the number of input nodes for the expert networks by exploring them individually, i.e., independent of the hierarchical structure. We tested expert networks with 5, 10, 25, 50, 100, and 150 input features. In each case we used twice the number of input nodes for the hidden layer and all three feature selection methods. The best result was obtained using 25 input features with 50 nodes in the hidden layer.

Having determined the optimal number of inputs, next we explored the effect of the size of the hidden layer for our networks. (Note that so far we have only explored the simple strategy of having twice the input nodes in the middle layer.) Table 6.2 shows the variations in performance with different sizes of the middle layer. In this case all networks have 25 inputs and a single output node. The best performance is obtained with expert networks that have 6 nodes in the hidden layer. The difference between 6 and 10 nodes is relatively small. Observe also that the flat neural networks have their best performance when the number of nodes in the hidden layer is 6. This is not surprising since a smaller hidden layer tends to produce a better generalization of each category. We ran similar experiments on the gating networks varying the size of the middle layer (6, 10, 25, 50) and found that the best size of the hidden layer was 25 (Table 6.3). However, the difference between these runs is very small.

In the following sections we present results using neural networks with 25 input nodes, 6 hidden nodes and 1 output node for the experts and 25 input nodes, 25 hidden nodes and 1 output node for the gating networks.

6.6.2 Comparing the Category Zoning Methods

As explained in chapter 5.2, we use category zoning techniques to select the training set for each expert and gate. In particular we explore two different types of

Table 6.1: Effect of the number of input nodes to the gating networks and the feature selection methods for the expert networks[1]

Gating Networks[2]	Expert Networks[3]		
# of inputs	Corr. Coef.	Odds Ratio	Mutual Inf.
5	0.4455	0.4531	0.4589
10	0.4604	0.4695	0.4712
25	**0.4984**	**0.4961**	**0.4956**
50	0.4894	0.4903	0.4900
100	0.4894	0.4929	0.4840
150	0.4827	0.4890	0.4850
Flat[4]	0.4449	0.4488	0.4548

[1] Performance is measured in macro-averaged F_1 on the HD-49 set.

[2] The feature selection method for the gating networks is correlation coefficient.

[3] All expert networks have 25 input features selected by the indicated feature selection method, 50 nodes in the middle layer, and 1 output.

[4] The last row shows performance of the flat classifier.

Table 6.2: Effect of the number of hidden nodes on the expert networks[1]

# of hidden nodes	Flat NN[2]	HME[2]
6	0.5033	0.5241
10	0.4807	0.4975
25	0.4320	0.4824
50	0.4479	0.4867

[1] Performance is measured by macro-averaged F_1 on the HD-49 set.

[2] All expert networks have 25 input features selected with the correlation coefficient feature selection method.

Table 6.3: Effect of the number of hidden nodes on the gating networks[1]

# of hidden nodes	HME[2,3]
6	0.5219
10	0.5202
25	0.5241
50	0.5158

[1] Performance is measured by macro-averaged F_1 on the HD-49 set. nodes.
[2] The gates have 25 inputs selected with correlation coefficient.
[3] All expert networks have 25 input features and 6 hidden.

category zones: the centroid-based category zone and the Knn-based category zone. We compare these two zoning strategies with respect to their zone sizes as well as classifier performance.

The centroid-based category zone generates zones that have at least $10,000$ examples. For our training set of 103 categories we found that this type of category zone has an average size of $10,027$ with a maximum of $10,778$ while 75% of the zones are below $10,010$. The Knn-based category zone generate zones with sizes proportional to the number of positive examples in the category. We found that "compact" categories have in general small category zones. We explored different values of K (10, 50, 100, and 200). Small values of K are problematic for low frequency categories because they tend to generate a very small training set that the neural network overfits easily. Thus we settled for K=200 because it produces category zones large enough for the rare categories as well as zones of reasonable size for the more frequent categories. For our 103 categories we found that the average size of the Knn-based category zone is $6,098$ examples with a maximum of $35,917$, and a minimum of 200. 75% of the category zones generated by this method are below $6,814$. As mentioned before the category zone for a gate is the union of the individual category zones of its descendants in the hierarchy.

To measure the impact of each zoning method we trained both gating and expert networks with the documents of the corresponding zones. The categorization results on the test set are shown in Table 6.4. There are small differences in performance across zones for classifiers in the high frequencies (HD-49) set and for those classifiers in the medium frequencies (HD-28) set. However, these differences are not statistically significant.

The low frequency categories (HD-26) show a statistically significant difference in favor of the centroid-based zones for both flat and hierarchical classifiers. A detailed analysis shows that the high value of this difference is due in part to the contribution of some categories that have one or zero examples in the test set. For these categories the function becomes more of a hit or miss function[5]. Since these category zone training sets have at least $10,000$ examples the classifier learns to reject most of the documents and in consequence it gets an F_1 value of 1.0 for most of them. In

[5]Categories with no examples in the test set will have $F_1 = 0$ if a document is assigned to the class or $F_1 = 1$ if no documents are assigned to the class.

Table 6.4: Comparison of categorization performance using the centroid-based and Knn-based category zones[1]

	centroid-based zone		Knn-based zone	
	Flat NN	HME	Flat NN	HME
HD-49	0.5033	0.5241	0.5042	0.5150
HD-28	0.3589	0.4304	0.3613	0.4159
HD-26	0.5653	0.5794	0.4599	0.4828
HD-119	0.4797	0.5126	0.4542	0.4798

[1] Values are macro-averaged F_1 over the respective set of categories on the test set ($50,216$ documents).

contrast, the Knn-based zones for low frequency categories generate a smaller zone and the neural networks trained with them tend to assign the category to at least a few documents. The classifiers trained with centroid-based zones outperform the Knn-based classifiers on 13 categories, while the classifiers trained on Knn-based zones outperform the centroid-based classifiers only on 5 categories (in the remaining 8 categories there is no difference between them).

On the whole set HD-119 the classifiers trained with centroid-based zones outperform those trained with the Knn-based zones. However this difference is statistically significant only for the HME classifiers. Given our results with these two zoning methods we suggest that centroid-based category zones are the most appropriate for training hierarchical classifiers that span categories of varying frequencies. The same conclusion may be made for flat classifiers but with somewhat reduced confidence.

6.6.3 HME, Flat NN, and Optimized Rocchio Classifiers

Table 6.5 shows the performance of our flat neural network, the HME model and the optimized Rocchio classifiers, all trained using the centroid-based zoning method and features selected using correlation coefficient. The HME classifier consistently outperforms the flat neural network classifier in all category sets. The difference is statistically significant for HD-49, HD-28 and HD-119. Another important feature that we must point out is that our HME model has lower variance in performance in all the category sets. This result confirms the theoretical claim by Jordan and Jacobs that soft splitting is a variance reduction method [60]. This is in general a desirable property of a classifier since this indicates performance that is more stable across categories. Comparing the HME against the optimized Rocchio classifier we note that Rocchio significantly outperforms the HME on the HD-49 and HD-28 categories while the HME significantly outperforms Rocchio for the HD-26 categories. However, there is no significant difference between both classifiers on the HD-119 set. We have to admit that the good performance of the Rocchio classifier may in part be due to the particular combination of category zoning and feature selection methods used for our classifiers. To remind the reader, first a centroid-based category zone is identified for each category. This zone of documents is then used for feature selection. Although this approach is a "filter" approach the specifics of the centroid-based category zoning technique introduces a bias that may favor the Rocchio classifier. Our Rocchio results also emphasize the conclusion by Schapire *et al.* that when the Rocchio classifier is

Table 6.5: Comparison between the flat NN, HME NN and the optimized Rocchio classifiers

		Flat NN	HME NN	Opt-Rocchio
Macro	HD-49	0.5033	0.5241	0.5491
Avg F_1	HD-28	0.3589	0.4304	0.5176
	HD-26	0.5653	0.5794	0.4524
	HD-119	0.4797	0.5126	0.5161
Variance	HD-49	0.02589	0.02298	0.02470
	HD-28	0.06746	0.06773	0.05467
	HD-26	0.17879	0.14583	0.16775
	HD-119	0.08000	0.06754	0.06877

properly trained, it performs as well as other methods [118]. This result is in contrast with the performance of Rocchio classifiers observed by other researchers [65, 78, 136].

A detailed analysis of the behavior of the HME with respect to the flat neural network shows that the thresholds for the expert nodes of the hierarchical classifier is less than or equal to the thresholds for the networks of the flat classifier in 95 of the 103 categories. This is an expected result because the intermediate layers perform a pre-filtering of "bad candidate texts" hence the experts receive a smaller number of examples in the hierarchical approach. Since the optimization process sets these thresholds to maximize the F_1 values in the training set, when the bad matches have been filtered the algorithm is able to set a lower threshold that increases the number of true positives without significantly increasing the number of false positives. The idea is to have a hierarchy that is good at filtering out false positives.

Table 6.6 shows the number of documents that pass through each gate in the test set. The number of documents in the test collection is 50,216. The root node filters out most of the documents since only $9,238$ pass through it. Observe that gates 1.11, 1.11.8, and 1.11.9 allow a big portion of the documents to pass through to the lower levels. This is because they contain two of the categories with the highest number of training examples ("Coronary Diseases", and "Myocardial Infarction"). We expected this to harm the performance of the rest of the categories in these subnodes but in practice this did not happen. For example, only 2 of the 8 categories in the Coronary Diseases subtree (not shown in the table) have a slightly lower performance in the HME model than in the flat classifier.

6.6.4 Comparing Results with other Published Works

The OHSUMED collection has been used by very few researchers for text categorization. Moreover, to the best of our knowledge only two studies have used its entire set of $14,000$ MeSH categories [67, 135]. The main reason for this is that many text categorization methods do not scale to such a large dataset. Yang [135], Lewis *et al.* [78], and Lam and Ho [65] have published results using the subset of categories from the "Heart Diseases" sub-tree (HD-119). This has become a standard set for comparing results for text categorization in the OHSUMED collection. However, when reading these three works carefully we found that each paper uses a different test set and report results on a different number of categories.

Lewis *et al.* use the set of $183,229$ documents from 1987 to 1990 for training and all the $50,216$ documents from the year 1991 as a test set. Our experiments follow

Table 6.6: Number of documents that pass each HME NN gate in the test set

Level 1		# of doc. ≥ threshold
(root) Heart Diseases		9,238
Level 2	Level 3	
1.1 Arrhythmia		1,464
	1.1.1 Heart Block	176
	1.1.2 Pre-Excitation Syndromes	45
	1.1.3 Tachycardia	583
1.2 Endocarditis		293
	1.2.4 Endocarditis, Bacterial	216
1.4 Heart Defects Congenital		1,187
	1.3.5 Heart Septal Defects	277
	1.3.6 Transposition of Great Vessels	57
1.6 Heart Failure,Congestive		1,591
1.7 Heart Rupture		377
1.8 Heart Valve Diseases		1,592
1.9 Myocardial Diseases		4,281
	1.9.7 Cardiomyopathy, Hypertrophic	102
1.10 Pericarditis		447
1.11 Myocardial Ischemia		5,769
	1.11.8 Coronary Diseases	3,343
	1.11.9 Myocardial Infarction	2,570

exactly this partition with a further reduction for training (using zoning techniques) but the test set is the same. Table 6.7 shows that our flat neural network model performs at the same level for HD-49 and slightly worse for HD-28, as the Exponentiated Gradient (EG) algorithm in Lewis *et al.* [78]. EG is the second ranked and the top ranked algorithm for HD-49 and HD-28 respectively in [78]. (Note that the last three rows of the table show results from our work that are most comparable). The HME model on the other hand shows a performance slightly lower (4.7%) than the top ranked Widrow-Hoff (WH) for HD-49 but significantly higher than the best (10.2%) performance for HD-28. Both the flat NN and HME outperform the Rocchio classifier reported as a baseline by Lewis *et al.* [78]. Interestingly, our optimized Rocchio classifier performs at the same level as WH for HD-49 but significantly better than all other classifiers for HD-28.

Yang [135] conducts a very different experiment by reducing the collection to only those documents that are positive examples of the categories of the HD-119. This limits the training set to $12,284$ documents and the test set to $3,760$ documents. She explains that the reason for such reduction is the scalability of the LLSF method which needs to compute a Singular Values Decomposition, a procedure that cannot be performed efficiently on a large matrix. However, this simplification creates a partition of the OHSUMED collection that is structurally different from the one originally proposed by Lewis *et al.* In order to explore differences we used our trained classifiers on this reduced test set (with the same thresholds as before). The results obtained are 0.525, 0.521 and 0.530 for the flat neural networks, the HME and optimized Rocchio respectively. Observe that for our hierarchical model, Yang's reduction is equivalent to having a perfect classifier at the root node of the tree and thus using only gates at levels 2, 3 and 4. We then ran a second experiment where although we did not retrain the neural networks or Rocchio classifiers on the reduced training set, threshold selection was done only on the set of $12,824$ positive examples in the training set (instead of using the whole set of $183,229$ documents of the training set) and the gates in levels 2, 3 and 4 were used. The results for our flat neural networks, HME and optimized Rocchio classifiers on this reduced subset are 0.564, 0.557, and 0.558 respectively, scores which are significantly higher than those we found using the Lewis *et al.* partition. These scores are about the same as the ones reported by Yang for the LLSF and ExpNet classifiers, and significantly above Yang's baseline STR classifier. Interestingly the HME model does not outperform the flat neural network model in this constrained experiment. We looked closely at each category and found that this was due to the fact that our originally trained gates discard documents that are relevant to their descendants which then impacts the final performance of the classifier. Although performance may be improved if we train the gates using only the set of positive examples[6], we believe that our original categorization task is more realistic since we include both positive and negative examples in the test set.

Lam and Ho [65] report results of experiments using the Generalized Instance

[6]In fact, in experimentation with Knn zoning which uses a smaller training set that more closely resembles the positive examples, HME was better than the flat neural network classifier.

Table 6.7: Performance comparison between the flat NN, HME and Rocchio classifiers, and classifiers published in other works

	Yang [135]	Lewis *et al.* [78]	
Method	HD-119	HD-49	HD-28
LLSF[1]	0.55	–	–
ExpNet[1]	0.54	–	–
STR[1]	0.38	–	–
Rocchio[2]	–	0.44	0.33
EG[2]	–	0.50	0.39
WH[2]	–	0.55	0.39
Flat NN	0.564	0.503	0.358
HME	0.557	0.524	0.430
Opt-Rocchio	0.558	0.549	0.518

[1] Yang [135].
[2] Lewis *et al.* [78].

Set (GIS) algorithm. They use the documents from 1991 and take the first 33,478 documents for training and the last 16,738 documents for testing. In contrast to Yang's reduction, this is more consistent with the partition proposed by Lewis *et al.* because it retains all documents from the collection (not just the positive examples). Lam and Ho report results using micro averaged BEP on the set of 84 categories that have at least one example in both training and test sets. We tested our previously trained HME model in this reduced test set and obtained a micro-averaged BEP value of 0.502 which is significantly lower than their performance of 0.572 for their GIS algorithm with Rocchio generalization. In future research we will train and test our HME model using their data set.

Joachims [57] has also published results for the OHSUMED collection using support vector machines. His work uses the first $20,000$ documents of the year 1991 dividing it into two sets of $10,000$ documents each that are used for training and testing respectively. He reports impressive results but his text categorization task is very different from the ones in the previously discussed works. Joachims assumes that if a category in the UMLS tree is assigned then its more general category in the hierarchy is also present. Although this is similar to our definition of gates, the difference is that he uses only the high level disease categories. This simplifies the categorization task considerably and probably explains the good results obtained in the reported SVM experiments. The focus on general disease categories alone prevents comparison of Joachims' results with any of the previously published results. We did not run our experiments limited to high level disease categories. However, we consider that the use of SVM can be a good choice for building an architecture like the one we have proposed, and we plan to explore this in our future research.

We believe that the combination of category zoning and feature selection gives a significant boost to Rocchio performance. Optimized versions of the Rocchio classifier in previous work have focused on query zoning and dynamic feedback optimization [118]. We are not aware of any previous work on reducing the set of features used in the centroid vector for text categorization purposes. Observe that our feature selection method favors features indicative of the presence of the category and discards features that indicate the absence of the category. Feature selection has an important impact because similarity computation between the document and the centroid vector are made only on the subspace formed by the selected features.

6.7 Comparison with Related Work in Hierarchical Categorization

We now focus on methods exploring the hierarchical structures of classification schemes.

Koller and Sahami [63, 111] proposed a hierarchical approach that trains independent Bayesian classifiers for each node of the hierarchy. The classification scheme then starts at the root and greedily selects the best link to a second level classifier. This process is repeated until a leaf is reached or until no child node is a good candidate. Observe that their method selects a single path (the one with highest probability) and assigns all the categories in the path to the document. According to their approach, the hierarchical structure is used as a filter that activates only a single best classification path. Also errors in classification at the higher levels are irrecoverable in the lower levels. They tested their results in the Reuters collection defining as higher nodes in the hierarchy those categories that subsume other categories. Although similar in spirit, our approach differs in the classification assignment model. We separate the identification of general concepts from the assignment of general categories. Our approach also activates more than a single path in the hierarchy in contrast to their "the winner takes all" approach. There is also an obvious difference in the machine learning algorithm since they use Bayesian classifiers while we use neural networks. However in future work we plan to explore Bayesian classifiers within the HME approach.

Ng *et al.* [96] build their hierarchical classifier using perceptrons. Each node of the hierarchical tree is represented by a perceptron. They distinguish two types of nodes, leaf nodes and non-leaf nodes. They apply this to the Reuters corpus where the categories reflect a geographical/topical hierarchy. Their hierarchy has as a first level all the possible countries, and for each country different topics are defined, *e.g.*, economics and politics. The leaf nodes are specific categories of the second level (*e.g.*, for economics they have communications, industry, *etc.*). The hierarchical classifier receives a document and checks whether it belongs to any of the first level nodes (the root node only connects to the different country nodes). If the tested document activates any of the first level nodes, then the descendant categories of that node are tested recursively. If at any of the non-leaf nodes the process finds that none of its children is a good candidate, then the categorization stops at that branch of the recursion. The output of the classifier is the final set of leaf nodes reached in the recursion (zero, one or more). This is similar to the *Pachinko machine* proposed by Koller and Sahami [63] but with multiple outputs instead of a single output. Our approach is similar to Ng *et al.* [96] in that we also use a top-down approach. The difference is in the type of classifier. We use non-linear classifiers in each node while they use linear classifiers. Although the combination of the linear classifiers in the hierarchy creates a non-linear classifier some studies have shown that this covers a limited number of non linear problems[7] [60, 128]. Our approach also

[7]This is due to the fact that the high level nodes can only create linear boundaries between adjacent expert regions in the input space.

differs significantly in the way feature selection and subset training selection is done. Although their experiments also use correlation coefficient for feature selection their results for the hierarchical classifier are well below other methods that use the Reuters collection. They report F_1 values of 0.52 for the best automatic feature selection method and 0.728 for the manually selected features (which is considerable lower than 0.85 [136]). Their best results with the hierarchical classifier are obtained using manually selected features. We believe that their good results may be a consequence of manual feature selection. Furthermore, our approach based on category zones combined with exploiting the hierarchy is more robust and allows us to get results similar to some of the best methods reported in the literature.

McCallum and his collaborators have been working in text categorization specifically targeting the problem of classifying web pages [85]. Their approach is based on Bayesian classifiers. They use the hierarchical classification structure to improve the accuracy of Bayesian classifiers using a statistical technique called shrinkage that smoothes parameter estimates of a child node with its parent in order to obtain more robust estimates. The Bayesian classification schemes involves estimating the parameters of the model from the training collection and then applying the shrinkage method to improve the estimates using the predefined vocabulary hierarchy. The classification of the test set is performed by computing the posterior probability of each class given the words observed in the test document, and selecting the class with the highest probability. Their experiments show that shrinkage improves the performance when the training data is sparse, reducing the classification error by up to 29%. Our approach is totally different from McCallum's approach in terms of the classification method used, as well as the assumptions in the categorization task. Observe that their approach assumes that a document belongs to a single category and their model reflects it by selecting the most probable classification.

Mladenić [90] also explored hierarchical structures using the Yahoo! hierarchy to classify web pages. Her approach builds a Bayesian classifier. For each node in the subject hierarchy a classifier is induced. To train each of the non-leaf classifiers a set of positive examples is defined as all the positive examples of the node (the intermediate nodes are also valid categories) plus the positives examples of any descendants. These examples are weighted according to their position in the tree. The classification process works as described before for the Bayesian classifiers with the set of categories with predicted probability ≥ 0.95 are assigned. Our approach differs from Mladenić's work in terms of the classifier algorithms used, as well as the way in which the hierarchy is used for feature selection. Her main effort is in creating a weighting scheme for combining probabilities obtained at different nodes of the tree.

The work on topic spotting by Wiener *et al.* [131] inspired us to try our approach using HME. They have recently published a sequel to their work applied to hierarchical classifiers [129]. They use a meta-topic network using a two level hierarchy for the Reuters collection and present results that are competitive with those of other methods. Our work differs from theirs in the definition of the gates (our gates behave like binary filters while their meta-topic network helps more to weight the contribution of the experts). Our work also extends the hierarchical model to more than two levels an idea that is suggested but not developed in [129]. Finally, our evaluation on the OHSUMED collection instead of the Reuters collection allows us to test the benefit of exploiting a real hierarchical classification scheme.

We think that our main contribution is a general method for combining the hierarchical structure of a classification with feature selection and category zones. The zones contain small but optimal subsets of documents that yield features suitable for training neural networks. Our approach shows that even with a set of only 25 features per node, we can get results that are comparable with other methods that use larger feature sets[8].

[8]Most [96, 131, 129] have used feature sets of 200 or more features per category. In [78] the authors have used all features present in the collection.

CHAPTER 7
EXPERIMENTS WITH THE HME MODEL BUILT ON OTHER CLASSIFIERS

Our main thesis in this dissertation is that it is possible to exploit the hierarchical structure of a classification scheme via the HME classifier. The focus of our research is primarily on the hierarchical architecture of the classifier and not on the particular classifier technology used at each node. Although we have thus far chosen to employ backpropagation neural nets for these nodes, any other type of decision making module may be used instead to implement the HME model. We pursue this idea by exploring two linear classifiers Widrow-Hoff (WH) and Exponentiated Gradient (EG). We use the standard implementations of these algorithms as described in section 3.3. We use the same set of features selected for the neural networks and Rocchio classifiers using correlation coefficient.

7.1 Experiments with the HME Model Using Linear Classifiers

In this set of experiments we use the Widrow-Hoff (WH) classifier and the Exponentiated Gradient (EG) classifier as the nodes of the HME model. For the WH classifier we initialize the classifier with the vector $(0, 0, \ldots, 0)$ and for the learning rate we use $\eta = 1/(4X^2)$ where X is an upper bound on $\| \mathbf{x} \|$ for all instances $\mathbf{x}$. We choose this value as the maximum value present in the training collection. This

Table 7.1: Comparison between the flat EG and WH classifiers and the corresponding HME classifiers using EG and WH gating nodes

		Flat EG	HME EG	Flat WH	HME WH
Macro Avg F_1	HD-49	0.3982	0.4473	0.5363	0.5429
	HD-28	0.2996	0.3618	0.4584	0.4786
	HD-26	0.1473	0.2520	0.3603	0.3550
	HD-119	0.3081	0.3748	0.4707	0.4779
Variance	HD-49	0.03023	0.02473	0.02794	0.02478
	HD-28	0.05631	0.05014	0.06814	0.05225
	HD-26	0.04923	0.08750	0.15426	0.12802
	HD-119	0.05171	0.05276	0.07421	0.06275

learning rate has bee proposed by Kivinen and Warmuth [61] and has been used by Lewis *et al.* [78] for text categorization.

For the EG algorithm we initialize the vector $\mathbf{w}$ as $(1/m, \ldots, 1/m)$ (where m is the number of features of th input vector). The learning rate $\eta = 2/(3R^2)$ (where $max_j(x_{ij}) - min_j(x_{ij}) \leq R$ is also selected based on Kivinen and Warmuth [61] and on Lewis *et al.* [78].

In order to obtain a binary classification we use the same thresholding technique (as with the neural networks) based on optimizing the F_1 measure on the training set. We trained linear classifiers for both gates and experts nodes in the HME. The performance of these HME classifiers on the test set is presented in Table 7.1.

The results show that the HME approach improves performance compared to the flat classifier for almost all cases, with the exception of the HD-26 for WH where

Table 7.2: Comparison between the flat WH and Rocchio classifiers and the corresponding hybrid HME classifiers using NN gating nodes

		Flat WH	HME NN&WH	Flat Rocchio	HME NN&Rocchio
Macro	HD-49	0.5363	0.5524	0.5491	0.5380
Avg F_1	HD-28	0.4584	0.4719	0.5176	0.5269
	HD-26	0.3603	0.3562	0.4524	0.3998
	HD-119	0.4707	0.4810	0.5161	0.5001
Variance	HD-49	0.02794	0.02549	0.02470	0.02232
	HD-28	0.06814	0.06329	0.05467	0.05026
	HD-26	0.15426	0.11009	0.16775	0.13767
	HD-119	0.07421	0.06217	0.06877	0.06100

the performance drops slightly. This difference is statistically significant for all sets of categories in EG and not significant for all sets in WH. There is also a variance reduction effect when we use the HME model with the exception of the low frequency categories HD-26 and in the whole set HD-119 in the EG algorithm.

The Widrow-Hoff (WH) hierarchical classifier shows a high performance on the HD-49 classes and is very similar to the 0.55 reported by Lewis *et al.* [78][1]. This is consistent with Kivinen and Warmuth [61] expected square loss formula which suggests that the WH algorithm should perform well when we can find a vector that fits the data and the number of training examples is large in comparison to the length of the fitting vector and the representative document vectors. When we compare the performance of the WH classifier against the EG classifier we note a significant difference in performance that favors the WH algorithm. This is consistent with the

[1]We should note that Lewis *et al.* use all the features available in the training documents and the full training set for their experiments

results reported by Lewis *et al.* [78].

When we compare these linear classifiers against the results obtained for the neural network based HME (Table 6.5) we see that the neural networks model perform significantly better than the HME EG model in all four sets of categories. HME NN outperform the HME WH classifiers on the global subset HD-119. However, the source of this difference is due to the significantly higher performance of the neural networks on the low frequency (HD-26) categories. In fact, the WH algorithm outperforms the neural networks on the HD-49 and HD-28 sets of categories.

Table 7.2 shows two hybrid HME models where the gates are neural networks and the experts are WH and Rocchio classifiers. This combination of classifiers shows that using different classifiers in the gates and experts could contribute to improve results compared to using a single classifier. However, because the WH classifier does not perform well in the low frequency categories the improvements are only noticeable on the high level categories. Observe also that the use of neural networks in the gating nodes improves performance for the HD-49, HD-28 and HD-119 sets of categories. For the Hybrid HME NN-Rocchio classifier we observed that improvements in performance occur only in the HD-28 categories and drops slightly for the other three sets of categories but non of the differences are statistically significant. This was quite surprising because we expected to get similar improvements as with the hybrid HME NN-WH model. Performing a detailed examination of the results at each category we found that performance improves in 20 categories and drops in 19 categories and there are no changes in the remaining 74 categories.

Tables 7.3 and 7.4 show the behavior of the gates implemented in the HME-WH and HME-EG linear classifiers respectively. Compared to the gates implemented in the HME-NN algorithms they seem to behave very differently. The first level node is more aggressive in the HME linear classifiers than in the HME neural networks. However, the middle level nodes seem to be accepting a significantly larger number of documents than the neural networks. This explains why the HME models implemented with the linear classifiers yields a lower performance increment than the HME models using neural networks gating nodes.

7.2 Ensembles of Classifiers and the HME Model

The results obtained in the previous section points toward the fact that combining classifiers that use different approaches improves results in categorization. This technique has been studied by several researchers in machine learning and the classifier obtained is called an *ensemble* [7, 98, 132]. An ensemble is defined as a set of individually trained classifiers whose outputs are combined to obtain the final classification of a new instance.

Ensembles have also been applied in text categorization by several researchers [71, 117, 118, 121, 144]. Larkey and Croft reported that the combination of multiple classifiers yields significant performance improvements [71]. Yu and Liddy [144] have also used an ensemble of classifiers generated by a neuro-genetic algorithm. Their ensemble computes the weighted sum of the scores of the neuro-genetic classifiers to

Table 7.3: Number of documents that pass each HME WH gate in the test set

Level 1		# of doc ≥ threshold
(root) Heart Diseases		8,463
Level 2	Level 3	
1.1 Arrhythmia		2,868
	1.1.1 Heart Block	817
	1.1.2 Pre-Excitation Syndromes	406
	1.1.3 Tachycardia	2,106
1.2 Endocarditis		408
	1.2.4 Endocarditis, Bacterial	380
1.4 Heart Defects Congenital		2,065
	1.3.5 Heart Septal Defects	1,361
	1.3.6 Transposition of Great Vessels	261
1.6 Heart Failure,Congestive		4,020
1.7 Heart Rupture		2,866
1.8 Heart Valve Diseases		2,702
1.9 Myocardial Diseases		6,195
	1.9.7 Cardiomyopathy, Hypertrophic	1,720
1.10 Pericarditis		1,784
1.11 Myocardial Ischemia		6,853
	1.11.8 Coronary Diseases	5,912
	1.11.9 Myocardial Infarction	3,329

Table 7.4: Number of documents that pass each HME EG gate in the test set

Level 1		# of doc. ≥ threshold
(root) Heart Diseases		8,133
Level 2	Level 3	
1.1 Arrhythmia		3,430
	1.1.1 Heart Block	1,228
	1.1.2 Pre-Excitation Syndromes	763
	1.1.3 Tachycardia	1,374
1.2 Endocarditis		1,591
	1.2.4 Endocarditis, Bacterial	1,264
1.4 Heart Defects Congenital		2,598
	1.3.5 Heart Septal Defects	1,535
	1.3.6 Transposition of Great Vessels	1,760
1.6 Heart Failure,Congestive		5,146
1.7 Heart Rupture		3,669
1.8 Heart Valve Diseases		3,942
1.9 Myocardial Diseases		5,669
	1.9.7 Cardiomyopathy, Hypertrophic	1,541
1.10 Pericarditis		5,411
1.11 Myocardial Ischemia		6,949
	1.11.8 Coronary Diseases	6,053
	1.11.9 Myocardial Infarction	3,770

obtain the final classification.

As pointed out by Opitz and Maclin [98] combining classifiers helps only when the classifiers disagree amongst themselves on how to classify a new document. Two main approaches have been proposed to generate classifiers that differ in their classification decisions (even though they might use the same learning algorithm): bagging and boosting. The bagging method, proposed by Breiman [7], is based on the statistical “bootstrap” method proposed by Ephron [27, 28]. In this case several training sets are generated by sampling with replacement the original training set. These training sets are used to train several classifiers which are used to build an ensemble by computing the weighted average of their classification scores. Boosting, proposed by Schapire [34, 116], includes methods that are used to improve performance in a series of classifiers. The training set selected for the next classifier in the series emphasizes the use of those examples that were classified incorrectly by the previous classifier in the series. The method selects a probabilistic sample (from the training set) that gives a higher probability to the misclassified examples in the previous step. Then the current classifier is trained on this set and the process is repeated until the classifier achieves perfect classification of the training set or until it reaches a termination condition.

We have trained different classifiers (neural networks, WH, EG and Rocchio) using different category zones and feature selection methods. Given the results obtained in our previous experiments we decided to built ensembles of classifiers that combine the neural network classifiers that we obtained with the two different category zones, the two linear classifiers and the Rocchio classifier. We use the binary decision of each classifier to compute a simple majority voting to obtain the final decision of the ensemble. Thus the approach that we follow to build the ensemble is closer to the bagging approach but used different zones, features and classification algorithms instead of using bootstrapping.

We explore the idea of ensembles within both the flat and the HME architectures. Each flat classifier ensemble is built as a combination of the 5 classifiers mentioned above. The ensemble makes the classification decision via a majority vote as depicted in Figure 7.1. For the HME ensemble we decided to built a hierarchy of flat ensembles. That is, each expert node is represented by its corresponding flat ensemble while the gates are neural networks trained with the centroid based zones. Figure 7.2 illustrates the HME ensemble for a gate with a couple of expert nodes. In other words, the HME model fits the hierarchical structure of the classification on top of ensembles of flat classifiers. Another alternative that we did not explore here is to build an ensemble of gates and then combine them with the ensembles of experts using the HME model. A third potential alternative is to consider each HME model as a full classifier and combine alternative HME classifiers into an ensemble of HME classifiers.

Table 7.5 shows on the top part the performance of the individual classifiers and their corresponding hierarchical versions. The bottom part of the table shows

Table 7.5: Performance of ensembles of classifiers using a simple majority

	HD-49	HD-28	HD-26	HD-119
Flat NN_C	0.5033	0.3589	0.5652	0.4797
Flat NN_{Knn}	0.5042	0.3613	0.4599	0.4542
Flat WH	0.5364	0.4585	0.3603	0.4708
Flat EG	0.3983	0.2996	0.1474	0.3081
Flat Rocchio	0.5491	0.5176	0.4524	0.5161
HME NN_C	0.5241	0.4304	0.5794	0.5126
HME NN_{Knn}	0.5150	0.4155	0.4828	0.4798
HME WH	0.5429	0.4786	0.3550	0.4780
HME EG	0.4473	0.3618	0.2521	0.3748
	Ensembles			
Flat NN_C+NN_{Knn}+WH	0.5559	0.4264	0.4982	0.5062
Flat NN_C+NN_{Knn}+Rocc	0.5566	0.4674	0.5430	0.5289
Flat NN_C+NN_{Knn}+WH+Rocc	0.5752	0.4829	0.2803	0.4757
Flat NN_C+NN_{Knn}+WH+Rocc+EG	0.5656	0.4482	0.2866	0.4633
HME NN_C+NN_{Knn}+WH	0.5594	0.4215	0.5046	0.5081
HME NN_C+NN_{Knn}+Rocc	0.5545	0.4463	0.5590	0.5262
HME NN_C+NN_{Knn}+WH+Rocc	0.5637	0.4503	0.2931	0.4646
HME NN_C+NN_{Knn}+WH+Rocc+EG	0.5675	0.4390	0.3088	0.4672

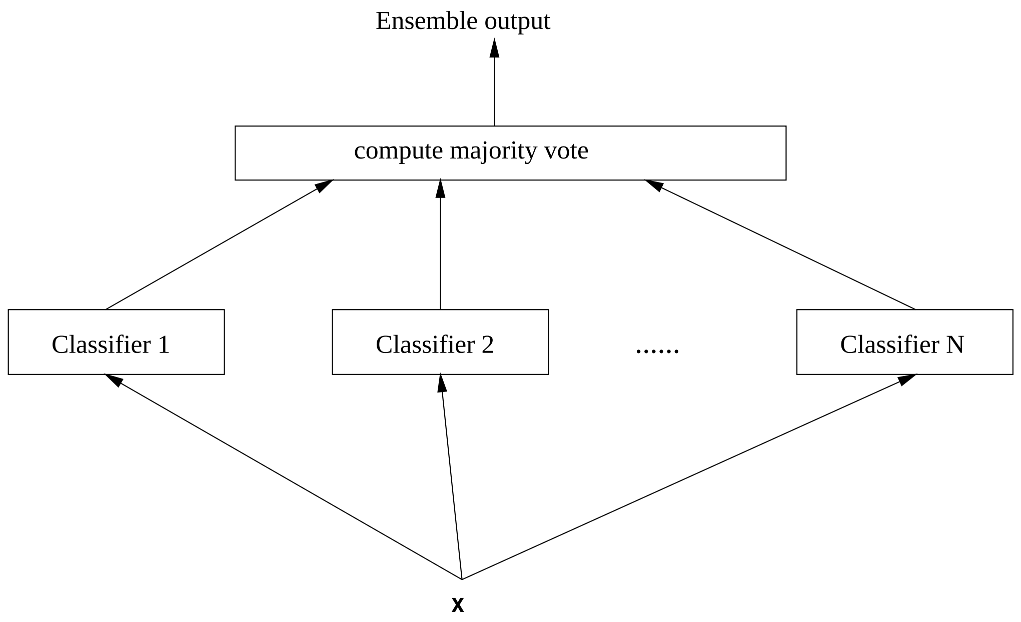

Figure 7.1: Ensemble classifier

the four flat ensembles and their respective hierarchical versions. All these ensembles include the two neural networks trained with correlation coefficient features using the centroid-based zone (NN_C) and Knn-based zone (NN_{Knn}) and one or more of the linear classifiers (WH, EG, Rocchio). Observe that all the ensembles except one have an odd number of classifiers so that the majority decision is always reached. For the ensemble that combines the two neural networks, WH and Rocchio classifiers we break ties by favoring the assignment of the category.

Table 7.5 shows that the use of ensembles improves categorization performance for the HD-49 categories (all the ensembles significantly outperform any of the individual classifiers for this set of categories). For the medium frequency categories (HD-28) the performance of the ensembles is better than the corresponding individual classifier

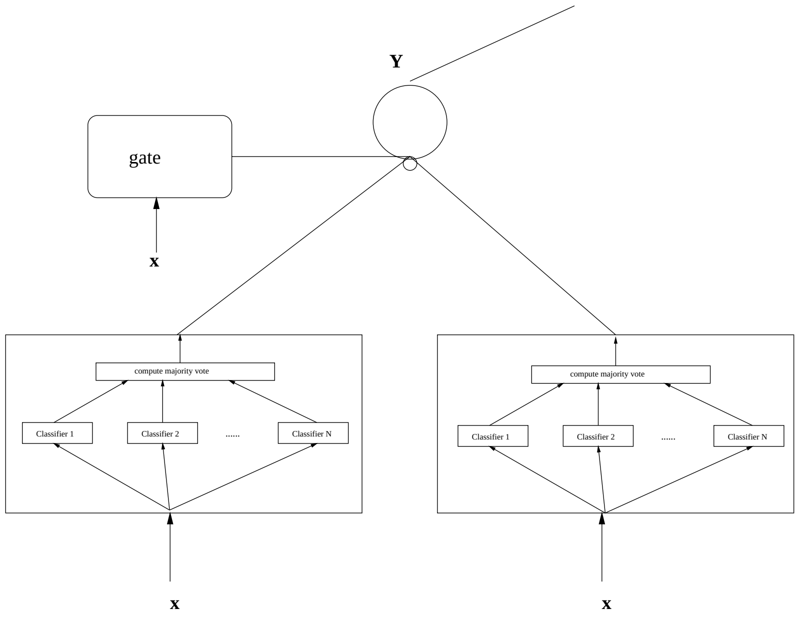

Figure 7.2: Gate with two experts in the HME ensemble classifier

except for the Rocchio classifier. For the low frequency categories (HD-26) the neural networks trained on the centroid-based zones outperform all the other classifier. The performance of the ensembles seem to decrease when more linear classifiers are added to the ensemble. This indicates that possibly the linear classifiers agree in many of these classes and the decisions tend to be dominated by them. The combination of NN_C+NN_{Knn}+WH+Rocchio classifiers appear to be the best for HD-49 and HD-28 while NN_C+NN_{Knn}+Rocchio is the best for HD-26 and HD-119.

For the HME ensemble the performance for the high frequency categories HD-49 is consistently better than the performance of the corresponding HME models without ensembles. However, for the HD-28, HD-26 and HD-119 the results are mixed. We also note that for the ensembles the hierarchical versions do not outperform the corresponding flat ensembles. Further investigation of these differences is left to future research where we will more fully study alternative ensemble architectures.

Comparing the results of the ensembles with other published results we note that the best ensemble classifier (NN_C+NN_{Knn}+WH+Rocchio) outperforms the best classifier reported by Lewis *et al.* [78] by 5.2% on the HD-49 categories and by 23.8% on the HD-28 categories. The corresponding hierarchical version of the Ensemble classifier also outperforms Lewis *et al.* by 3.0% on the HD-49 categories and by 15.5% on the HD-28 categories.

This concludes our presentation of experiments using the HME model. We have shown that in general the use of the HME model improves performance of an equivalent non hierarchical classifier. The next chapter presents our conclusions and future lines of research in this area.

CHAPTER 8
CONCLUSIONS AND FUTURE WORK

This dissertation explored the use of hierarchical classifiers for text categorization. Our main goal was to exploit the underlying hierarchical structure of the classification scheme using a variation of the Hierarchical Mixtures of Experts proposed by Jordan and Jacobs [60]. This final chapter summarizes our findings and presents the conclusions derived from our research. We conclude by presenting our plans for future research.

8.1 Conclusions

Our experiments on the OHSUMED collection showed that the hierarchical model improves categorization performance with respect to an equivalent flat classifier that uses neural network, Exponentiated Gradient or Widrow-Hoff classifiers. However, in some cases using the HME in conjunction with other classifiers such as Rocchio classifiers or ensembles reduces classification performance slightly but not significantly. Despite the fact that the hierarchical model does not show consistent improvements in classification performance for all methods, its major merit could be the solution of the scalability problem for text categorization with a large classification scheme. Our experiments show that the high level nodes of the hierarchical model are able to filter out documents that are unlikely to belong to any of the categories of its descendants. A system based on the HME model would activate only those subsets of categories that are likely to be relevant to a new document that needs to be classified. This approach could prove useful to implement categorization using large classification schemes such as the UMLS ($350,000$ concepts), MeSH ($165,000$ concepts), International Patent Classification (IPC) ($60,000$ concepts) or the European Patent Office Classification System (ECLA) ($100,000$ concepts).

The use of zoning methods for training set selection and their combination with feature selection methods has proven to be an effective way to reduce training time for neural networks without sacrificing performance. Our classifiers trained on these reduced sets achieve performance comparable to other published results that use a larger training set. This leads us to conclude that a good set of positive examples combined with closely related negative examples can be sufficient to properly train a machine learning method for text categorization.

The study of feature selection also showed us that a carefully selected set of input features allows us to achieve competitive results thus minimizing training time for the neural network classifier. From our work we can conclude that localized feature selection combined with measures to address problems caused by low frequency terms can succeed in identifying the best set of features for individual classifiers.

In general it appears that HME had greater success with HD-49, i.e. the 49 categories that had at least 75 positive examples in training. The most difficult was the HD-26 categories that have less than 15 examples were performance varied widely. The larger improvement of the HME with respect to the flat model are obtained for the HD-28 categories that have between 15 and 74 examples in the training set. This seems to indicate that the HME model is more dependent upon the availability of positive examples.

Our work also confirmed the thesis that a properly trained Rocchio algorithm can achieve performance competitive with more complex methods. In particular we have discovered that the Rocchio algorithm benefits from careful feature selection, an aspect that has not been investigated before. This approach, combined with the category zone, produces a classifier that performs at the same level as the best published text categorization methods. In fact, our Rocchio classifier is the best of the individual flat classifiers and only the ensembles achieve better performance than our baseline Rocchio classifier.

We also explored the use of ensembles for text categorization. Our results confirm that an ensemble of different classifiers improves categorization performance They yield the best performance of all the classifiers tested in this work which is higher than the best published results for this collection (5.2% for HD-49 and 23.8% for HD-28). However, the version of hierarchical ensembles does not show the same improvements of categorization performance with respect to the flat classifiers. In fact performance decreases slightly but not significantly in any of the four groups of categories that we studied but still higher than the best published results for this collection (3.0% for HD-49 and 15.5% in HD-28). The construction of hierarchical classifiers that use ensembles is still an open question that we intend to address in our future research.

In terms of efficiency, working with a large data set like OHSUMED for text categorization allowed us to realize how important it is to use distributed processing. The training phase of the neural networks is a computing intensive task that requires a considerable time to complete. We created a client server application that allows the distribution of training over a pool of computers. The server controls the progress of the training and the clients simply ask for allocation of new assignments. This tool reduced significantly the time needed for training the neural networks from 77 hours (3.25 days) to an overnight process (it takes between 8 to 10 hours to complete a full training cycle of the 103 experts and 11 gates using 15 computers).

8.2 Future Research

There are several open issues of research that result from this work. One important aspect is to see if a wrapper approach towards feature selection would make a difference to the relative standing of HME with neural network nodes. As mentioned previously correlation coefficient filters combined with category zoning may favor Rocchio classifiers because this is one of the best way to improve the Rocchio algorithm. A second line of research is to expand our variety of feature types. Instead of limiting our features to stems derived from single words, we could explore phrases as features.

It may also prove beneficial to focus more closely on the particular challenges offered by the low frequency classifiers, i.e., the categories with few positive examples. In particular it may be worthwhile to explore the value of bootstrapping methods such as boosting for these categories.

This thesis has just began the exploration of ensembles both within and external to the HME model. Results obtained are significantly intriguing to warrant further exploration on alternative ensemble architectures and key concepts of bias versus variance across classifiers in the ensemble.

On a somewhat different direction we also would like to explore the automatic categorization of heterogeneous collections such as web pages using existing classification hierarchies such as Yahoo!. What intrigues us about heterogeneous collections is that unlike the OHSUMED dataset, structural characteristics such as document size and vocabulary are likely to vary widely across the documents.

In conclusion, I hope that with this thesis I have provided the text categorization community reasonable responses to our initial questions regarding hierarchical models, while also identifying future areas of investigation.

APPENDIX A
HEART DISEASES SUB TREE OF THE UMLS METATHESAURUS

1 C0018799 Heart Diseases

1.1 C0003811 Arrhythmia

1.1.1 C0003813 Arrhythmia, Sinus

1.1.2 C0004238 Atrial Fibrillation

1.1.3 C0004239 Atrial Flutter

1.1.4 C0006099 Bradycardia

1.1.5 C0015374 Extrasystole

1.1.6 C0018794 Heart Block

1.1.6.1 C0001396 Adams-Stokes Syndrome

1.1.6.2 C0006384 Bundle-Branch Block

1.1.6.3 C0037188 Sinoatrial Block

1.1.7 C0023976 Long QT Syndrome

1.1.8 C0032915 Pre-Excitation Syndromes

1.1.8.1 C0024054 Lown-Ganong-Levine Syndrome

1.1.8.2 C0032916 Pre-Excitation, Mahaim-Type

1.1.8.3 C0043202 Wolff-Parkinson-White Syndrom

1.1.9 C0037052 Sick Sinus Syndrome

1.1.10 C0039231 Tachycardia

1.1.10.1 C0039236 Tachycardia, Paroxysmal

1.1.10.2 C0039240 Tachycardia, Supraventricular

1.1.10.2.1 C0039232 Tachycardia, Atrioventricular Nodal Reentry

1.1.10.2.2 C0039234 Tachycardia, Ectopic Atrial

1.1.10.2.3 C0039235 Tachycardia, Ectopic Junctional

1.1.10.2.4 C0039238 Tachycardia, Sinoatrial Nodal Reentry

1.1.10.2.5 C0039239 Tachycardia, Sinus

1.1.10.2.6 C0040479 Torsades de Pointes

1.1.10.2.7 C0078888 Idioventricular Rhythm

1.1.10.3 C0042514 Tachycardia, Ventricular

1.1.11 C0042510 Ventricular Fibrillation

1.1.12 C0206068 Parasystole

1.2 C0007093 Carcinoid Heart Disease

1.3 C0007166 Cardiac Output, Low

1.4 C0007177 Cardiac Tamponade

1.5 C0014118 Endocarditis

1.5.1 C0014121 Endocarditis, Bacterial

1.5.1.1 C0014122 Endocarditis, Subacute Bacterial

1.6 C0018789 Heart Aneurysm

1.7 C0018790 Heart Arrest

1.7.1 C0085298 Death, Sudden, Cardiac

1.8 C0018798 Heart Defects, Congenital

1.8.1 C0003492 Aortic Coarctation

1.8.2 C0009995 Cor Triatriatum

1.8.3 C0010074 Coronary Vessel Anomalies

1.8.4 C0010334 Crisscross Heart

1.8.5 C0011813 Dextrocardia

1.8.6 C0013274 Ductus Arteriosus, Patent

1.8.7 C0013481 Ebstein's Anomaly

1.8.8 C0013743 Eisenmenger Complex

1.8.9 C0018816 Heart Septal Defects

1.8.9.1 C0003516 Aortopulmonary Septal Defect

1.8.9.2 C0014116 Endocardial Cushion Defects

1.8.9.3 C0018817 Heart Septal Defects, Atrial

1.8.9.3.1 C0024164 Lutembacher's Syndrome

1.8.9.3.2 C0041022 Trilogy of Fallot

1.8.9.4 C0018818 Heart Septal Defects, Ventricular

1.8.10 C0023569 Levocardia

1.8.11 C0039685 Tetralogy of Fallot

1.8.12 C0040761 Transposition of Great Vessels

1.8.12.1 C0013069 Double Outlet Right Ventricle

1.8.13 C0041207 Truncus Arteriosus, Persistent

1.9 C0018800 Heart Hypertrophy

1.9.1 C0007193 Cardiomyopathy, Congestive

1.9.2 C0149721 THypertrophy, Left Ventricular

1.9.3 C0162770 Hypertrophy, Right Ventricular

1.10 C0018802 Heart Failure, Congestive

1.10.1 C0013405 Dyspnea, Paroxysmal

1.10.2 C0013608 Edema, Cardiac

1.11 C0018809 Heart Neoplasms

1.12 C0018813 Heart Rupture

1.12.1 C0018814 Heart Rupture, Post-Infarction

1.13 C0018824 Heart Valve Diseases

1.13.1 C0003504 Aortic Valve Insufficiency

1.13.2 C0003505 Aortic Valve Prolapse

1.13.3 C0003507 Aortic Valve Stenosis

1.13.4 C0018808 Heart Murmurs

1.13.5 C0026266 Mitral Valve Insufficiency

1.13.6 C0026267 Mitral Valve Prolapse

1.13.7 C0026269 Mitral Valve Stenosis

1.13.8 C0034088 Pulmonary Valve Insufficiency

1.13.9 C0034089 Pulmonary Valve Stenosis

1.13.10 C0040961 Tricuspid Valve Insufficiency

1.13.11 C0040962 Tricuspid Valve Prolapse

1.13.12 C0040963 Tricuspid Valve Stenosis

1.13.13 C0079485 Heart Valve Prolapse

1.14 C0027049 Myocardial Diseases

1.14.1 C0007192 Cardiomyopathy, Alcoholic

1.14.2 C0007194 Cardiomyopathy, Hypertrophic

1.14.2.1 C0003500 Aortic Subvalvular Stenosis

1.14.2.2 C0034084 Pulmonary Subvalvular Stenosis

1.14.3 C0007196 Cardiomyopathy, Restrictive

1.14.4 C0007930 Chagas Cardiomyopathy

1.14.5 C0014117 Endocardial Fibroelastosis

1.14.6 C0014183 Endomyocardial Fibrosis

1.14.7 C0022541 Kearns Syndrome

1.14.8 C0027055 Myocardial Reperfusion Injury

1.14.9 C0027059 Myocarditis

1.15 C0031039 Pericardial Effusion

1.16 C0031046 Pericarditis

1.16.1 C0031048 Pericarditis, Constrictive

1.16.2 C0031049 Pericarditis, Tuberculous

1.17 C0032319 Pneumopericardium

1.18 C0032805 Postpericardiotomy Syndrome

1.19 C0034072 Pulmonary Heart Disease

1.20 C0035439 Rheumatic Heart Disease

1.21 C0042512 Ventricular Outflow Obstruction

1.22 C0085128 Cardiac Output, High

1.23 C0151744 Myocardial Ischemia

1.23.1 C0010068 Coronary Disease

1.23.1.1 C0002962 Angina Pectoris

1.23.1.1.1 C0002963 Angina Pectoris, Variant

1.23.1.1.2 C0002965 Angina, Unstable

1.23.1.1.3 C0206064 Syndrome X

1.23.1.2 C0010051 Coronary Aneurysm

1.23.1.3 C0010054 Coronary Arteriosclerosis

1.23.1.4 C0010072 Coronary Thrombosis

1.23.1.5 C0010073 Coronary Vasospasm

1.23.2 C0027051 Myocardial Infarction

1.23.2.1 C0036980 Shock, Cardiogenic

1.23.3 C0206146 Myocardial Stunning

APPENDIX B

LIST OF STOPWORDS

a	am	asking	between	concerning
a’s	among	associated	beyond	consequently
able	amongst	at	both	consider
about	an	available	brief	considering
above	and	away	but	contain
according	another	awfully	by	containing
accordingly	any	b	c	contains
across	anybody	be	c’mon	corresponding
actually	anyhow	became	c’s	could
after	anyone	because	came	couldn’t
afterwards	anything	become	can	course
again	anyway	becomes	can’t	currently
against	anyways	becoming	cannot	d
ain’t	anywhere	been	cant	definitely
all	apart	before	cause	described
allow	appear	beforehand	causes	despite
allows	appreciate	behind	certain	did
almost	appropriate	being	certainly	didn’t
alone	are	believe	changes	different
along	aren’t	below	clearly	do
already	around	beside	co	does
also	as	besides	com	doesn’t
although	aside	best	come	doing
always	ask	better	comes	don’t

done	ex	getting	hence	ie
down	exactly	given	her	if
downwards	example	gives	here	ignored
during	except	go	here's	immediate
e	f	goes	hereafter	in
each	far	going	hereby	inasmuch
edu	few	gone	herein	inc
eg	fifth	got	hereupon	indeed
eight	first	gotten	hers	indicate
either	five	greetings	herself	indicated
else	followed	h	hi	indicates
elsewhere	following	had	him	inner
enough	follows	hadn't	himself	insofar
entirely	for	happens	his	instead
especially	former	hardly	hither	into
et	formerly	has	hopefully	inward
etc	forth	hasn't	how	is
even	four	have	howbeit	isn't
ever	from	haven't	however	it
every	further	having	i	it'd
everybody	furthermore	he	i'd	it'll
everyone	g	he's	i'll	it's
everything	get	hello	i'm	its
everywhere	gets	help	i've	itself

j	look	namely	nowhere	out
just	looking	nd	o	outside
k	looks	near	obviously	over
keep	ltd	nearly	of	overall
keeps	m	necessary	off	own
kept	mainly	need	often	p
know	many	needs	oh	particular
knows	may	neither	ok	particularly
known	maybe	never	okay	per
l	me	nevertheless	old	perhaps
last	mean	new	on	placed
lately	meanwhile	next	once	please
later	merely	nine	one	plus
latter	might	no	ones	possible
latterly	more	nobody	only	presumably
least	moreover	non	onto	probably
less	most	none	or	provides
lest	mostly	noone	other	q
let	much	nor	others	que
let's	must	normally	otherwise	quite
like	my	not	ought	qv
liked	myself	nothing	our	r
likely	n	novel	ours	rather
little	name	now	ourselves	rd

re	seen	somewhere	thats	this
really	self	soon	the	thorough
reasonably	selves	sorry	their	thoroughly
regarding	sensible	specified	theirs	those
regardless	sent	specify	them	though
regards	serious	specifying	themselves	three
relatively	seriously	still	then	through
respectively	seven	sub	thence	throughout
right	several	such	there	thru
s	shall	sup	there’s	thus
said	she	sure	thereafter	to
same	should	t	thereby	together
saw	shouldn’t	t’s	therefore	too
say	since	take	therein	took
saying	six	taken	theres	toward
says	so	tell	thereupon	towards
second	some	tends	these	tried
secondly	somebody	th	they	tries
see	somehow	than	they’d	truly
seeing	someone	thank	they’ll	try
seem	something	thanks	they’re	trying
seemed	sometime	thanx	they’ve	twice
seeming	sometimes	that	think	two
seems	somewhat	that’s	third	u

un	w	whereafter	wonder
under	want	whereas	would
unfortunately	wants	whereby	would
unless	was	wherein	wouldn’t
unlikely	wasn’t	whereupon	x
until	way	wherever	y
unto	we	whether	yes
up	we’d	which	yet
upon	we’ll	while	you
us	we’re	whither	you’d
use	we’ve	who	you’ll
used	welcome	who’s	you’re
useful	well	whoever	you’ve
uses	went	whole	your
using	were	whom	yours
usually	weren’t	whose	yourself
uucp	what	why	yourselves
v	what’s	will	z
value	whatever	willing	zero
various	when	wish	
very	whence	with	
via	whenever	within	
viz	where	without	
vs	where’s	won’t	

REFERENCES

[1] I. Androutsopoulos, J. Koutsias, K. Chandrinos, and C. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In N. Belkin, P. Ingwersen, and M.-K. Leong, editors, *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–167, Athens, Grece, July 2000. ACM, ACM Press.

[2] C. Apté, F. J. Damerau, and S. M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.

[3] B. Berlin. *Ethnological Classification: Principles of Categorization of Plants and Animals in Traditional Societies.* University Press, Princeton, NJ, 1992.

[4] P. Biebricher, N. Fuhr, G. Knorz, G. Lustig, and M. Schwantner. The automatic indexing system AIR/PHYS. From research to application. In Y. Chiaramella, editor, *Proceedings of SIGIR-88, 11th ACM International Conference on Research and Development in Information Retrieval*, pages 333–342, Grenoble, FR, 1988. ACM Press, New York, US. Reprinted in Karen Sparck Jones and Peter Willett (eds.), "Readings in Information Retrieval", Morgan Kaufmann, San Francisco, US, 1997, pp. 513–517.

[5] M. Blosseville, G. Hebrail, M. Montell, and N. Penot. Automatic document classification: natural language processing and expert system techniques used together. In N. J. Belkin, P. Ingwersen, and A. M. Pejtersen, editors, *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 51–57, Kobenhavn, DK, 1992. ACM Press, New York, US.

[6] H. Borko and M. Bernick. Automatic document classification. *Journal of the Association for Computing Machinery*, 10(2):151–161, 1963.

[7] L. Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996.

[8] L. Breiman, J. H. Friedman, R. A. Olshen, , and C. J. Stone. *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA, 1984.

[9] J. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fogelman-Soulie and J. Hérault, editors, *Neuro-computing: Algorithms, Architectures, and Applications*. Springer-Verlag, New York, NY, 1989.

[10] C. Buckley and G. Salton. Optimization of relevance feedback weights. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 351–357, New York, NY, July 1995. ACM press.

[11] W. Buntine. Introduction to ind and recursive partitioning. Technical report, RIACS/NASA Ames Research Center, September 1991.

[12] C. Cleverdon. Optimizing convenient on-line access to bibliographic databases. *Information Services and Use*, 4(1):37–47, 1984.

[13] W. W. Cohen. Learning to classify English text with ILP methods. In L. De Raedt, editor, *Advances in inductive logic programming*, pages 124–143. IOS Press, Amsterdam, NL, 1995.

[14] W. W. Cohen. Text categorization and relational learning. In A. Prieditis and S. J. Russell, editors, *Proceedings of ICML-95, 12th International Conference on Machine Learning*, pages 124–132, Lake Tahoe, US, 1995. Morgan Kaufmann Publishers, San Francisco, US.

[15] W. W. Cohen and H. Hirsh. Joins that generalize: text classification using WHIRL. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of KDD-98, 4th International Conference on Knowledge Discovery and Data Mining*, pages 169–173, New York, US, 1998. AAAI Press, Menlo Park, US.

[16] W. W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 307–315, Zürich, CH, 1996. ACM Press, New York, US. An extended version appears as [17].

[17] W. W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems*, 17(2):141–173, 1999.

[18] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[19] L. Cranor and B. A. LaMancchia. Spam! *Communications of the ACM*, 41(8):74–83, 1998.

[20] M. Craven, D. DiPasquo, D. Freitag, A. K. McCallum, T. M. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 509–516, Madison, US, 1998. AAAI Press, Menlo Park, US. An extended version appears as [21].

[21] M. Craven, D. DiPasquo, D. Freitag, A. K. McCallum, T. M. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, 118(1-2):69–113, 2000.

[22] R. M. Creecy, B. M. Masand, S. J. Smith, and D. L. Waltz. Trading MIPS and memory for knowledge engineering: classifying census returns on the Connection Machine. *Communications of the ACM*, 35(8):48–63, 1992.

[23] I. Dagan, Y. Karov, and D. Roth. Mistake-driven learning in text categorization. In C. Cardie and R. Weischedel, editors, *Proceedings of EMNLP-97, 2nd Conference on Empirical Methods in Natural Language Processing*, pages 55–63, Providence, US, 1997. Association for Computational Linguistics, Morristown, US.

[24] R. Dattola. A fast algorithm for automatic classification. *Journal of Library Automation*, 2:31–48, 1969.

[25] S. T. Dumais and H. Chen. Hierarchical classification of Web content. In N. J. Belkin, P. Ingwersen, and M.-K. Leong, editors, *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, Athens, GR, 2000. ACM Press, New York, US.

[26] S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In G. Gardarin, J. C. French, N. Pissinou, K. Makki, and L. Bouganim, editors, *Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management*, pages 148–155, Bethesda, US, 1998. ACM Press, New York, US.

[27] B. Efron. Bootstrap methods: Another look to the jacknife. *Annals of Statistics*, 17:1–26, 1979.

[28] B. Efron and R. Tibshirani. *An Introduction to Bootstrap.* Chapman and Hall, 1993.

[29] R. Ellen. *Indigenous classifications.* Butterword, 1990.

[30] W. K. Estes. *Classification and Cognition.* Oxford University Press, New York: NY, 1994.

[31] H. Fangmeyer and G. Lustig. The EURATOM automatic indexing project. In *Proceedings of the IFIP Congress (Booklet J)*, pages 66–70, Edinburgh, UK, 1968.

[32] B. Field. Towards automatic indexing: automatic assignment of controlled-language indexing and classification from free indexing. *Journal of Documentation*, 31(4):246–265, 1975.

[33] W. B. Frakes. Stemming algorithms. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, chapter 8, pages 131–160. Prentice Hall, Englewood Cliffs, NJ, 1992.

[34] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of he Thirteenth International Conference on Machine Learning*, pages 148–156, Bari, Italy, 1996.

[35] R. M. Frumkina and A. V. Mikhejev. *Meaning and Categorization.* Nova Science Publishers, New York: NY, 1996.

[36] N. Fuhr. A probabilistic model of dictionary-based automatic indexing. In *Proceedings of RIAO-85, 1st International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 207–216, Grenoble, FR, 1985.

[37] N. Fuhr, S. Hartmann, G. Knorz, G. Lustig, M. Schwantner, and K. Tzeras. AIR/X – a rule-based multistage indexing system for large subject fields. In A. Lichnerowicz, editor, *Proceedings of RIAO-91, 3rd International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 606–623, Barcelona, ES, 1991. Elsevier Science Publishers, Amsterdam, NL.

[38] N. Fuhr and G. Knorz. Retrieval test evaluation of a rule-based automated indexing (AIR/PHYS). In C. J. van Rijsbergen, editor, *Proceedings of SIGIR-84, 7th ACM International Conference on Research and Development in Information Retrieval*, pages 391–408, Cambridge, UK, 1984. Cambridge University Press.

[39] W. A. Gale, K. W. Church, and D. Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26(5):415–439, 1993.

[40] W. A. Gray and A. J. Harley. Computer-assisted indexing. *Information Storage and Retrieval*, 7(4):167–174, 1971.

[41] K. A. Hamill and A. Zamora. An automatic document classification system using pattern recognition techniques. In E. H. Brenner, editor, *Proceedings of ASIS-78, 41st Annual Meeting of the American Society for Information Science*, pages 152–155, New York, US, 1978. American Society for Information Science, Washington, US.

[42] K. A. Hamill and A. Zamora. The use of titles for automatic document classification. *Journal of the American Society for Information Science*, 33(6):396–402, 1980.

[43] P. J. Hayes and S. P. Weinstein. CONSTRUE/TIS: a system for content-based indexing of a database of news stories. In A. Rappaport and R. Smith, editors, *Proceedings of IAAI-90, 2nd Conference on Innovative Applications of Artificial Intelligence*, pages 49–66. AAAI Press, Menlo Park, US, 1990.

[44] H. Heaps. A theory of relevance for automatic document classification. *Information and Control*, 22(3):268–278, 1973.

[45] M. A. Hearst. Noun homograph disambiguation using local context in large corpora. In *Proceedings of the 7th Annual Conference of the University of Waterloo Center for the New Oxford English Dictionary*, pages 1–22, Oxford, UK, 1991.

[46] W. Hersh, C. Buckley, T. J. Leone, and D. Hickam. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In W. B. Croft and C. J. V. Rijsbergen, editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–201, Dublin, Ireland, July 1994. ACM, Springer-Verlag.

[47] W. Hoyle. Automatic indexing and generation of classification by algorithm. *Information Storage and Retrieval*, 9(4):233–242, 1973.

[48] D. Hull. *Information Retrieval Using Statistical Classification.* PhD thesis, Stanford University, 1994.

[49] D. A. Hull. Improving text retrieval for the routing problem using latent semantic indexing. In W. B. Croft and C. J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 282–289, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.

[50] D. A. Hull. The TREC-7 filtering track: description and analysis. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of TREC-7, 7th Text Retrieval Conference*, pages 33–56, Gaithersburg, US, 1998. National Institute of Standards and Technology, Gaithersburg, US.

[51] S. M. Humphrey. Medindex–the medical indexing expert system. In R. Aluri and D. E. Riggs, editors, *Expert Systems in Libraries*, chapter 14, pages 192–221. Ablex Publishing Co, Norwood:NJ, 1990.

[52] D. J. Ittner, D. D. Lewis, and D. D. Ahn. Text categorization of low quality images. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 301–315, Las Vegas, US, 1995.

[53] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In D. H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 143–151, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[54] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. Published in the "Lecture Notes in Computer Science" series, number 1398.

[55] T. Joachims. Estimating the generalization performance of a svm efficiently. Technical Report LS-8 Report 25, Universität Dortmund, Dortmund, Germany, December 1999.

[56] T. Joachims. Transductive inference for text classification using support vector machines. In I. Bratko and S. Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[57] T. Joachims. Estimating the generalization performance of a SVM efficiently. In P. Langley, editor, *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 431–438, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US.

[58] T. Joachims, D. Freitag, and T. M. Mitchell. WebWatcher: a tour guide for the Word Wide Web. In M. E. Pollack, editor, *Proceedings of IJCAI-97, 15th International Joint Conference on Artificial Intelligence*, pages 770–775, Nagoya, JP, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[59] G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, San Francisco, CA, 1994. AAAI, Morgan Kaufmann Publishers.

[60] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. Technical Report A.I. Memo No. 1440, Massachusetts Institute of Technology, Cambridge, MA, August 1993.

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

[62] P. H. Klingbiel. Machine-aided indexing of technical literature. *Information Storage and Retrieval*, 9(2):79–84, 1973.

[63] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In D. H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 170–178, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[64] S. L. Lam and D. L. Lee. Feature reduction for neural network based text categorization. In A. L. Chen and F. H. Lochovsky, editors, *Proceedings of DASFAA-99, 6th IEEE International Conference on Database Advanced Systems for Advanced Application*, pages 195–202, Hsinchu, TW, 1999. IEEE Computer Society Press, Los Alamitos, US.

[65] W. Lam and C. Y. Ho. Using a generalized instance set for automatic text categorization. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 81–89, Melbourne, AU, 1998. ACM Press, New York, US.

[66] W. Lam, K. F. Low, and C. Y. Ho. Using a Bayesian network induction approach for text categorization. In M. E. Pollack, editor, *Proceedings of IJCAI-97, 15th International Joint Conference on Artificial Intelligence*, pages 745–750, Nagoya, JP, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[67] W. Lam, M. E. Ruiz, and P. Srinivasan. Automatic text categorization and its applications to text retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 11(6):865–879, 1999.

[68] K. Lamberts. Process models of categorization. In K. Lamberts and D. Shanks, editors, *Knowledge, Concepts, and Categories*, chapter chap. 10, pages 371–403. MIT Press, 1997.

[69] L. S. Larkey. Automatic essay grading using text categorization techniques. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 90–95, Melbourne, AU, 1998. ACM Press, New York, US.

[70] L. S. Larkey. A patent search and classification system. In E. A. Fox and N. Rowe, editors, *Proceedings of DL-99, 4th ACM Conference on Digital Libraries*, pages 179–187, Berkeley, US, 1999. ACM Press, New York, US.

[71] L. S. Larkey and W. B. Croft. Combining classifiers in text categorization. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 289–297, Zürich, CH, 1996. ACM Press, New York, US.

[72] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In N. J. Belkin, P. Ingwersen, and A. M. Pejtersen, editors, *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 37–50, Kobenhavn, DK, 1992. ACM Press, New York, US.

[73] D. D. Lewis. Evaluating and optimizing autonomous text classification systems. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 246–254, Seattle, US, 1995. ACM Press, New York, US.

[74] D. D. Lewis. A sequential algorithm for training text classifiers: corrigendum and additional data. *SIGIR Forum*, 29(2):13–19, 1995.

[75] D. D. Lewis. Reuters-21578 text categorization test collection. Distribution 1.0, 1997. Available as `http://www.research.att.com/~lewis/reuters21578/README.txt`.

[76] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers.

In W. B. Croft and C. J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE. See also [74].

[77] D. D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, US, 1994.

[78] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 298–306, Zürich, CH, 1996. ACM Press, New York, US.

[79] J. B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistic*, 11:22–31, 1968.

[80] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2:159–165, April 1958.

[81] M. Maron. Automatic indexing: an experimental inquiry. *Journal of the Association for Computing Machinery*, 8(3):404–417, 1961.

[82] B. Masand, G. Linoff, and D. Waltz. Classifying news stories using memory-based reasoning. In N. J. Belkin, P. Ingwersen, and A. M. Pejtersen, editors, *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 59–65, Kobenhavn, DK, 1992. ACM Press, New York, US.

[83] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the 15th International Conference on Machine Learning*. AAAI, Morgan Kaufmann, July 1998.

[84] A. K. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In J. W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 350–358, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[85] A. K. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In J. W. Shavlik, editor,

Proceedings of ICML-98, 15th International Conference on Machine Learning, pages 359–367, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[86] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, London, England, 1989.

[87] T. Mitchell. Conditions for the equivalence of hierarchical and non-hierarchical bayesian classifiers. Technical report, Center for Automated Learning and Discovering, Carnegie Mellon University, 1998.

[88] T. Mitchell. *Machine Learning*. McGraw-Hill, 1998.

[89] D. Mladenić. *Machine Learning on non-homogeneous, distributed text data*. PhD thesis, University of Ljubljana, Faculty of Computer an Information Science, Ljubljana, Slovenia, 1998.

[90] D. Mladenić. *Machine Learning on non-homogeneous, distributed text data*. PhD thesis, J. Stefan Institute, University of Ljubljana, Ljubljana, SL, 1998.

[91] D. Mladenić. Turning YAHOO! into an automatic Web page classifier. In H. Prade, editor, *Proceedings of ECAI-98, 13th European Conference on Artificial Intelligence*, pages 473–474, Brighton, UK, 1998. John Wiley and Sons, Chichester, UK.

[92] I. Moulinier. A framework for comparing text categorization approaches. In *AAAI Spring Symposium on Machine Learning and Information Access*. Stanford University, March 1996.

[93] I. Moulinier and J.-G. Ganascia. Applying an existing machine learning algorithm to text categorization. In S. Wermter, E. Riloff, and G. Scheler, editors, *Connectionist, statistical, and symbolic approaches to learning for natural language processing*, pages 343–354. Springer Verlag, Heidelberg, DE, 1996. Published in the "Lecture Notes in Computer Science" series, number 1040.

[94] I. Moulinier, G. Raškinis, and J.-G. Ganascia. Text categorization: a symbolic approach. In *Proceedings of SDAIR-96, 5th Annual Symposium on Document Analysis and Information Retrieval*, pages 87–99, Las Vegas, US, 1996.

[95] G. L. Murphy and M. E. Lassaline. Hierarchical structure in concepts and the basic level of categorization. In K. Lamberts and D. Shanks, editors, *Knowledge, Concepts, and Categories*, chapter chap. 3, pages 93–132. MIT Press, 1997.

[96] H. T. Ng, W. B. Goh, and K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In N. J. Belkin, A. D. Narasimhalu, and P. Willett, editors, *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval*, pages 67–73, Philadelphia, US, 1997. ACM Press, New York, US.

[97] N. L. of Medicine. *Unified Medical Language System (UMLS) Knowledge Sources*. U.S. Department of Health and Human Services, National Institute of Health, National Library of Medicine, Bethesda, MD, 10th edition, 1999.

[98] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[99] H. Ragas and C. H. Koster. Four text classification algorithms compared on a Dutch corpus. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 369–370, Melbourne, AU, 1998. ACM Press, New York, US.

[100] J. Rennie and A. K. McCallum. Using reinforcement learning to spider the Web efficiently. In I. Bratko and S. Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 335–343, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[101] S. Robertson and D. A. Hull. The trec-9 filtering track final report. In *Proceedings of TREC-9, 9th Text Retrieval Conference*, Gaithersburg, US, 2000. National Institute of Standards and Technology.

[102] S. E. Robertson and P. Harding. Probabilistic automatic indexing by learning from human indexers. *Journal of Documentation*, 40(4):264–270, 1984.

[103] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, Englewood Cliffs: New Jersey, 1971.

[104] E. H. Rosch. Principles of categorization. In E. H. Rosch and B. Lloyd, editors, *Cognition and Categorization*, pages 27–48. Lawrence Erlbaum Associates, 1978.

[105] M. E. Ruiz and P. Srinivasan. Automatic text categorization using neural networks. In E. Efthimiadis, editor, *Proceedings of the 8th ASIS/SIGCR Workshop on Classification Research*, pages 59–72, Washington, US, 1997. American Society for Information Science, Washington, US.

[106] M. E. Ruiz and P. Srinivasan. Hierarchical neural networks for text categorization. In M. A. Hearst, F. Gey, and R. Tong, editors, *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 281–282, Berkeley, US, 1999. ACM Press, New York, US.

[107] D. E. Rumelhart, R. Durbin, R. Goldenand, and Y. Chauvin. Backpropagation: The basic theory. In M. C. Mozer and D. E. Rumelhart, editors, *Mathematical Perspectives on Neural Networks*, pages 533–566. Lawrence Associates, Hillsdale, NJ, 1996.

[108] D. E. Rumelhart and J. L. McClelland. *Parallel distributed processing: exploration in the microstructure cognition*, volume vols. 1 & 2. MIT Press, 1986.

[109] C. L. Sable and V. Hatzivassiloglou. Text-based approaches for the categorization of images. In S. Abiteboul and A.-M. Vercoustre, editors, *Proceedings of ECDL-99, 3rd European Conference on Research and Advanced Technology for Digital Libraries*, pages 19–38, Paris, FR, 1999. Springer Verlag, Heidelberg, DE. Published in the "Lecture Notes in Computer Science" series, number 1696. An extended version appears as [110].

[110] C. L. Sable and V. Hatzivassiloglou. Text-based approaches for non-topical image categorization. *International Journal of Digital Libraries*, 3(3):261–275, 2000.

[111] M. Sahami. *Using Machine Learning to Improve Information Access.* PhD thesis, Stanford University, Computer Science Department, 1998.

[112] M. Sahami, S. Dumais, D. Heckerman, and E. Horovitz. A basic approach to filtering junk e-mail. In *Learning for Text Categorization – Papers from the AAAI Workshop*, AAAI Technical Report WS-98-05, pages 55–62, Madison, Wisconsin, July 1998. AAAI, AAAI Press.

[113] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.

[114] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[115] G. Salton and M. J. McGill. *Introduction to modern information retrieval.* McGraw-Hill, 1983.

[116] R. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[117] R. E. Schapire and Y. Singer. BoosTexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[118] R. E. Schapire, Y. Singer, and A. Singhal. Boosting and Rocchio applied to text filtering. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 215–223, Melbourne, AU, 1998. ACM Press, New York, US.

[119] H. Schütze, D. A. Hull, and J. O. Pedersen. A comparison of classifiers and document representations for the routing problem. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 229–237, Seattle, US, 1995. ACM Press, New York, US.

[120] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 2002. Forthcoming.

[121] F. Sebastiani, A. Sperduti, and N. Valdambrini. An improved boosting algorithm and its application to automated text categorization. In A. Agah, J. Callan, and E. Rundensteiner, editors, *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pages 78–85, McLean, US, 2000. ACM Press, New York, US.

[122] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, Zurich, Switzerland, August 1996. ACM, ACM Press.

[123] A. Singhal, M. Mitra, and C. Buckley. Learning routing queries in a query zone. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25–32, Philadelphia, PA, July 1997. ACM, ACM Press.

[124] H. Taira and M. Haruno. Feature selection in SVM text categorization. In *Proceedings of AAAI-99, 16th Conference of the American Association for Artificial Intelligence*, pages 480–486, Orlando, US, 1999. AAAI Press, Menlo Park, US.

[125] K. Tzeras and S. Hartmann. Automatic indexing based on Bayesian inference networks. In R. Korfhage, E. Rasmussen, and P. Willett, editors, *Proceedings of*

SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval, pages 22–34, Pittsburgh, US, 1993. ACM Press, New York, US.

[126] C. van Rijsbergen. Automatic classification in information retrieval. *Drexel Library Quarterly*, 14:75–89, 1978.

[127] C. J. van Rijsbergen, D. J. Harper, and F. Porter, M. The selection of good search terms. *Information Processing & Management*, 17(2):77–91, 1981.

[128] S. R. Waterhouse. *Classification and regression using mixtures of experts.* PhD thesis, University of Cambridge, Cambridge, England, 1997.

[129] A. S. Weigend, E. D. Wiener, and J. O. Pedersen. Exploiting hierarchy in text categorization. *Information Retrieval*, 1(3):193–216, 1999.

[130] E. D. Wiener. A neural network approach to topic spotting in text. Master's thesis, Department of Computer Science, University of Colorado at Boulder, Boulder, US, 1995.

[131] E. D. Wiener, J. O. Pedersen, and A. S. Weigend. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332, Las Vegas, US, 1995.

[132] D. H. Wolpert. Stacked generalization. Technical Report LA-UR-90-3460, The Santa Fe Institute, Santa Fe, NM, 1993.

[133] J. Yang and V. Honovar. Feature subset selection using a genetic algorithm. In H. Liu and H. Motoda, editors, *Feature Extraction, Construction and Selection: A Data Mining Perspective*, chapter 8, pages 117–136. Kluwer Academic Publishers, 1998.

[134] Y. Yang. Expert network: effective and efficient learning from human decisions in text categorization and retrieval. In W. B. Croft and C. J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 13–22, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.

[135] Y. Yang. An evaluation of statistical approaches to MEDLINE indexing. In J. J. Cimino, editor, *Proceedings of AMIA-96, Fall Symposium of the American Medical Informatics Association*, pages 358–362, Washington, US, 1996. Hanley and Belfus.

[136] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69–90, 1999.

[137] Y. Yang, T. Ault, T. Pierce, and C. W. Lattimer. Improving text categorization methods for event tracking. In N. J. Belkin, P. Ingwersen, and M.-K. Leong, editors, *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 65–72, Athens, GR, 2000. ACM Press, New York, US.

[138] Y. Yang and C. G. Chute. An application of Least Squares Fit mapping to text information retrieval. In R. Korfhage, E. Rasmussen, and P. Willett, editors, *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval*, pages 281–290, Pittsburgh, US, 1993. ACM Press, New York, US. An extended version appears as [139].

[139] Y. Yang and C. G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3):252–277, 1994.

[140] Y. Yang and X. Liu. A re-examination of text categorization methods. In M. A. Hearst, F. Gey, and R. Tong, editors, *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49, Berkeley, US, 1999. ACM Press, New York, US.

[141] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In D. H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[142] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*. AAAI, AAAI Press, July 1997.

[143] Y. Yang and J. W. Wilbur. Using corpus statistics to remove redundant words in text categorization. *Journal of the American Society for Information Science*, 47(5):357–369, 1996.

[144] E. S. Yu and E. D. Liddy. Feature selection in text categorization using the Baldwin effect networks. In *Proceedings of IJCNN-99, 10th International Joint Conference on Neural Networks*, pages 2924–2927, Washington, DC, 1999. IEEE Computer Society Press, Los Alamitos, US.