USING MACHINE LEARNING TO IMPROVE INFORMATION ACCESS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Mehran Sahami
December 1998

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Daphne Koller
Computer Science Department
Stanford University
(Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Marti Hearst
School of Information Management and Systems
University of California at Berkeley

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Nils J. Nilsson
Computer Science Department
Stanford University

Approved for the University Committee on Graduate Studies:

Abstract

The explosion of on-line information has given rise to many query-based search engines (such as *Alta Vista*) and manually constructed topic hierarchies (such as *Yahoo!*). But with the current growth rate in the amount of information, query results grow incomprehensibly large and manual classification in topic hierarchies creates an immense information bottleneck. Therefore, these tools are rapidly becoming inadequate for addressing users' information needs.

In this dissertation, we address these problems with a system for topical information space navigation that combines the query-based and taxonomic approaches. Our system, named **SONIA** (**S**ervice for **O**rganizing **N**etworked **I**nformation **A**utonomously), is implemented as part of the Stanford Digital Libraries testbed. It enables the creation of dynamic hierarchical document categorizations based on the full-text of articles. Using probability theory as a formal foundation, we develop several Machine Learning methods to allow document collections to be automatically organized at a topical level. First, to generate such topical hierarchies, we employ a novel probabilistic clustering scheme that outperforms traditional methods used in both Information Retrieval and Probabilistic Reasoning. Furthermore, we develop methods for classifying new articles into such automatically generated, or existing manually generated, hierarchies. In contrast to standard classification approaches which do not make use of the taxonomic relations in a topic hierarchy, our method explicitly uses the existing hierarchical relationships between topics, leading to improvements in classification accuracy. Much of this improvement is derived from the fact that the classification decisions in such a hierarchy can be made by considering only the presence (or absence) of a small number of features (words) in each document.

The choice of relevant words is made using a novel information theoretic algorithm for feature selection. Many of the components developed as part of SONIA are also general enough that they have been successfully applied to data mining problems in different domains than text.

The integration of hierarchical clustering and classification will allow large amounts of information to be organized and presented to users in a individualized and comprehensible way. By alleviating the information bottleneck, we hope to help users with the problems of information access on the Internet.

Acknowledgments

During the time that I have been at Stanford, I have been extremely fortunate to have had the guidance, support, and friendship of a number of people who helped me grow academically and personally. This thesis would have been much different (or would not exist at all) if it were not for them.

The people who had the most direct impact on shaping the research in this thesis are the members of my thesis committee. I owe them all a tremendous intellectual debt. I would like to thank Nils Nilsson for introducing me to the topic of Machine Learning before I even started working on a Ph.D., and agreeing to be my advisor during the first two years of my graduate studies. I am always in awe of how quickly Nils can get to the real crux of AI problems, and I was equally amazed by how much patience he showed while guiding me through my first steps in Machine Learning. His ability to motivate students and genuinely care for their well-being is exceptional.

I met Marti Hearst during a summer internship at Xerox PARC. It was under her tutelage that I was introduced to Information Access and text categorization. That summer significantly changed the direction of my research, and, thankfully, Marti agreed to help me continue to explore this area by agreeing to join my thesis committee. I am indebted to her not only for her superb guidance, but also for always being a pleasure to work with. Her insightful comments and friendly demeanor always made our discussions very useful and fun. I learned a great deal from Marti. The opportunity to work with someone who has such a broad knowledge of Information Access was more than I could have asked for.

My thesis advisor, Daphne Koller, is the person who had the greatest role in my intellectual development during graduate school. It would be an understatement to say that she has been an exceptional advisor in all respects! Intellectually, she gave me an real appreciation for probability theory, and helped me develop a more rigorous approach toward research. Her high standards for research always pushed me to produce better work, and, thereby learn more. For that, I will always be grateful. As an advisor, she gave me the freedom to pursue the research areas that I was personally interested in, but still took a keen interest in helping me improve my work. During the past three years, Daphne has provided me with invaluable insights about research, teaching, and life in general. She is a great mentor and a good friend.

Many others also had in important role in helping me develop academically, and they all deserve my sincere thanks. I had the pleasure of working with Moises Goldszmidt while he sponsored me as a research associate at SRI. It was there that much of the clustering work in this thesis was done. Working with Moises was pure enjoyment—he challenged me intellectually, provided keen insights in our collaborations, and always kept me entertained with his sharp wit. One of the things I appreciate most about Moises is his great perspective on balancing personal and professional lives. Special thanks also go to Tom Mitchell whose encouragement (over a series of conversations in Crete and Italy, among other places) got me very excited about the possibilities of combining Machine Learning and Information Access. I also thank Tom for agreeing to be on my thesis committee, although time constraints precluded my being able to get his final approval on this document. I would like to thank Ross Shachter for introducing me to Bayesian Networks during my first year of graduate school, and his early words of encouragement that started me thinking more seriously about this area. Ross was also there to see me through to completion by serving as the chair of my orals committee.

Many others were also instrumental in getting me involved in projects that directly contributed to the work in this thesis, and I am indebted to them all. Yoav Shoham helped me get involved early in the Digital Libraries project at Stanford, and he also served as a member of my orals committee. Eric Saund sponsored an internship for me at Xerox PARC and mentored me during my introduction to text clustering and classification. Eric Horvitz invited me to spend a Fall at Microsoft Research, where I had a very fun and productive time working with him, Susan Dumais and David

Heckerman. I learned a great deal from the collaborations on all of these projects.

I have also benefited from the research groups I was fortunate enough to be involved with. I would like to thank many of the current and former members of the Nobots research group: Nir Friedman, George John, Ronny Kohavi, Pat Langley, Illah Nourbakhsh, Karl Pfleger, and Jeff Shrager. They all provided many enlightening discussions. George has been especially influential during my time at Stanford, and deserves special thanks. George and I worked together on several projects as undergraduates, taught classes together, and even ended up in the same research group (thanks to George introducing me to Nils). George is a good colleague and friend.

I also learned a great deal from the members of the Digital Libraries Project at Stanford: Marko Balabanovic, Michelle Baldonado, Sergey Brin, Steve Cousins, Hector Garcia-Molina, Luis Gravano, Scott Hassan, Steve Ketchpel, Andreas Paepcke, Larry Page, Terry Winograd, and the rest of the gang. I am also indebted to Salim Yusufali for helping implement a good deal of the infrastructure in SONIA. He is a fantastic developer.

More recently, I have gained a lot (both educationally and socially) from the members of DAGS (Daphne's Approximate Group of Students): Eric Bauer, Xavier Boyen, Urszula Chajewska, Raya Fratkina, Lise Getoor, Alex Kozlov, Uri Lerner, Ron Parr, Avi Pfeffer, and Simon Tong. My officemates Lise and Avi have been especially supportive. They are great people to talk to or just bounce ideas around with.

I would also like to thank many other colleagues with whom I have discussed machine learning and text categorization at length: Mark Craven, Doug Fisher, Rob Holte, Thorsten Joachims, David Karger, David Lewis, Andrew McCallum, Andrew Ng, Jan Pedersen, Hinrich Schuetze, Craig Silverstein, Sheila Tejada, Peter Turney, and Yiming Yang.

Besides research, another very important aspect of my life at Stanford has focused on teaching. Luckily, I had some exceptional teachers to teach me how to teach. Stuart Reges, my undergraduate advisor, introduced me to the joys of teaching through the CS198 undergraduate section leading program. He showed me that you can learn just as much from teaching a class as from taking one.

My most significant guidance in teaching (among other things) came from Eric

Roberts. I cannot thank Eric enough for everything that he has done for me since I was an undergraduate. Eric took me under his wing and on numerous occasions helped me develop as an educator. He gave me a real appreciate for curriculum development through all his hard work and dedication to students. I'm not sure I know anyone whose schedule is more full than his. Eric also instilled in me an appreciation for the ethical implications of work in Computer Science. He has been a truly outstanding educator and mentor, and a very caring friend. My life is better in many ways as a result of knowing Eric.

I also owe many thanks to the friends who helped keep me sane and happy during the past several years: Kevin Deeble, Stacey Doerr, Jonathan Eisenberg, Jason Holloway, Soraya Jenkins, Libusha Kelly, Andy Maag, Joe Matal, Sarah Richards, David Schuster, Marty Smith, Karin Tamerius, the fine residents of Casa Naranja '93-'97, and many other people that I don't have space to list. You know who you are.

I am especially deeply indebted to Sonia Giordani for always being with me through the good and bad times. She has been amazing in providing me with support and perspective during the sometimes trying process of writing a thesis. And, she stoically put up with this thesis taking longer to write than it was "supposed" to. She has greatly enriched my life intellectually, emotionally, and spiritually, and I cannot thank her enough for that. As you may have guessed, she is the namesake for the system described in this thesis. I would also like to thank her parents, Ezio and Anna Giordani, who have kindly treated by like their own son.

My brother Kamran Sahami and his partner Vicky Alten have helped keep my spirits up during the past few years by swapping ridiculous stories of graduate school experiences with me. This always helped keep the important things in perspective.

Finally, my deepest thanks and appreciation go to my parents, Mohammad and Soraya Sahami. Very early on, they instilled in me the importance of education. They have always given me their unconditional love, encouragement, and support, while giving me the freedom to pursue my own goals. I would never be here (in a variety of ways) if it was not for them. There is certainly no way that I can thank them enough for everything they have given me. This thesis is dedicated to them.

Contents

List of Tables

List of Figures

Contents

Part I

Preliminaries

Chapter 1

Introduction

In recent years, we have witnessed an immense growth in the availability of on-line information. Most notably, with the emergence of the World Wide Web, users now have access to thousands of information sources and untold millions of documents. Simultaneously, there has been a strong focus placed on Digital Libraries as a means of making such on-line information readily and easily available. Given that much of this information is textual in nature, the question arises of how access to so much information can be facilitated. Indeed, if we are not to drown in the growing sea of documents, it becomes necessary to build tools aimed at helping users find those documents that satisfy their *information needs*.

At a cursory level, the current methods for accessing large text collections (such as the Web) can be classified into two categories. The first group deals with retrieving documents from a collection in response to a user's query. Such methods are best exemplified by the **search engines** which are currently popular on the Web, such as *Alta Vista* [6, 151]. In this paradigm, a user is required to specify his or her information need in the form of a *query* which is then compared (typically at a simple keyword level) with documents in a collection to find those likely to be most related to the query and thus potentially relevant to the user. Alternatively, and more analogously to the traditional approach of libraries, on-line subject **taxonomies** (also called **directories**) for organizing information have been developed. These organizational schemes allow users to search manually for relevant documents by traversing a topic hierarchy, into which a collection is categorized. The taxonomy serves as a guide to help users home in on particularly useful documents. Such approaches have also gained great popularity on-line, as exemplified by the success of companies such as *Yahoo!* [173].

Unfortunately, these methods for accessing information are quickly becoming inadequate as the amount of on-line information continues to grow at an unprecedented rate. Methods that respond to a user's query with a simple list of documents quickly become unwieldy as the list of matching documents becomes incomprehensibly large. Even now, users often find themselves having to wade through several hundred documents returned in response to their queries; this situation will only get worse in the future.

On the other hand, current subject directory approaches suffer from the problem of having too little information available for users to peruse. Since the maintenance of such topic hierarchies currently requires the manual classification of new documents within the existing taxonomy, there is an immense bottleneck for keeping up with the amount of new information that is constantly becoming available. In fact, it has been estimated that, at the time of this writing, the World Wide Web contains over 300 million non-dynamically generated web pages [105]. In contrast, *Yahoo!*, the largest on-line directory, contains roughly 500,000 links to web pages in its topic hierarchy [160]. This staggering gap of almost three orders of magnitude between the amount of information available and the amount accessible via even the largest directory services show that manual classification into topic hierarchies simply can not keep up with the growth of the Web.

Realizing the shortcomings of current search engines and directory services, we consider how more sophisticated computational tools can be applied to help address the problems of information access. In order to pursue this goal, we first need a better understanding of the various challenges which arise in attempting to improve information access. We can then outline where the use of *intelligent* computation may provide a significant benefit. Thus, in the remainder of this chapter, we turn our attention to the goals of information access and subsequently give an outline of a system we have built for addressing some of these challenges. We also briefly review some of the previous research aimed at similar tasks, comparing and contrasting it with our approach. Finally, an overview of the component technologies encompassed by our system is provided. This overview also serves as a summary of the core technical contributions of this work, which are elaborated in depth in the subsequent chapters of this thesis.

1.1 Challenges of Information Access

As noted previously, users simply cannot manually look through the volumes of information that are available. Hence, it becomes clear that computational methods that help filter the store of information to allow users to more readily home in on documents of interest are needed. Over the past three decades, a growing body of research has emerged with the aim of addressing this issue. Often referred to as *Information Retrieval* (IR) or, more recently, *Information Access* (IA), this field has clearly defined a number of information access tasks which have become the focus of many recent research efforts in this area.

Several such information access tasks have been the focus of the *Text REtrieval Conference* (TREC) [65], which has in recent years served as a friendly competition among different research groups addressing information access problems. While, most recently, TREC has expanded to include such diverse tasks as cross-language and spoken document retrieval, we only presently consider the longer standing problems of *ad hoc retrieval*, as well as document *filtering* and *routing*. Moreover, moving beyond the realm of the tasks addressed in TREC, we also consider the new task of document *browsing*. Below we define and provide details on these different tasks.

1.1.1 Ad Hoc Retrieval

Given a large collection of documents, the **ad hoc retrieval** task centers on retrieving relevant documents in response to a user's query. Generally, systems built for addressing this task begin by *indexing* a collection of documents according to the words that the documents contain (referred to as an *inverted index* [51]). Such an index allows for the quick identification of those documents from the collection that contain any of the words in a user's query. After an initial set of matching documents is obtained, most retrieval systems will then rank these documents according to various criteria aimed at giving highest rank to the documents "most relevant" to the user's query.

Document ranking methods are often based on measures of how frequently query terms appear in each individual document as opposed to the document collection as a whole. Ranking measures, such as *TFIDF* weighting [145], usually weight each query term according to its rarity in the entire collection (often referred to as the *inverse document frequency*, or IDF) and then multiply this weight by the frequency of the corresponding query term in an individual document (referred to as the *term frequency*, or TF) to get an overall measure of that document's similarity to the user's query. The premise underlying such measures is that word frequency statistics provide a good measure for capturing a document's relevance to a user's information need (as articulated in his or her query). This conjecture generally has been borne out in previous TREC results, as many of the most successful retrieval systems have been based almost entirely on the analysis of term frequency statistics.

A more detailed overview of different ranking measures and their use in information retrieval systems can be found in [66]. Also, an excellent, detailed description of building complete systems for ad hoc retrieval as well as additional information on document ranking measures can be found in both [146] and [166]. A good history of the research development of SMART, one of the historically most important retrieval systems, is given in [144], which also outlines many of the classical research areas in building more effective retrieval systems.

1.1.2 Routing and Filtering

The related tasks of document routing and filtering can be thought of as a variation in the set-up of the ad hoc retrieval task. As mentioned previously, in ad hoc retrieval a static collection of documents is searched using dynamically generated user queries. In contrast, for both the routing and filtering tasks, a dynamic incoming document stream is matched against a static query, in order to capture new information relevant to a user's *persistent* information need. It is important to note here, however, that the user's standing information need may not be explicitly articulated as a query, although we may refer to it as such. Rather a user may express their information need simply via a *training set* of documents which may be categorized as either relevant or irrelevant by the user. In this regard, routing and filtering are analogous to task of classification, or *supervised* machine learning, which we discuss later in Section 3.2.1.

While closely related, the routing and filtering tasks have an important definitional difference. In document **routing**, we seek to produce a *ranked list* of documents according to their relevance to a standing query. In document **filtering**, however, we only make *binary decisions* as to whether or not given documents are relevant to a standing query. Consequently, any method for routing can be easily transformed into a method for filtering by simply introducing an appropriate threshold, and then marking as "relevant" only those documents whose measures of relevance are above this threshold. Likewise, any filtering method for classifying documents as being relevant or not, which also produces a confidence in its prediction of a document's relevance, can likewise be employed as a solution for the routing problem. Since, in the remainder of this work, any methods we describe for categorizing documents produce a real-valued confidence in their predictions, we conflate the routing and filtering problems as simply different instances of the document **classification** problem. More details on the document routing and filtering tasks, especially within the context of the TREC conference, can be found in [107] and [77].

1.1.3 Browsing

A final information access task which has become more pressing with the advent of the Web is that of browsing a document collection. We define **browsing** as: the task of becoming familiar with the contents of a collection in order to find individual or groups of documents that are relevant to an information need. In order to help facilitate browsing, it becomes necessary to provide infrastructure or computational tools that allow users to both get a better sense of the contents of a collection as well as quickly home in on documents of particular interest.

Simple examples of infrastructure that facilitate browsing are the hierarchical topic directories common in library settings (both the traditional paper as well as on-line "digital" variety) and the World Wide Web. Such directories capture a sense of the topics encompassed within a corpus and provide an explicit organizational scheme that enables users to find more quickly relevant documents. As users delve deeper into sub-categories within a directory, they quickly eliminate large numbers of irrelevant documents from consideration. Indeed, in order for on-line information to be truly useful, users must not only have some way of finding relevant information, but they must also have at their disposal means for filtering through the vast quantities of irrelevant material. Moreover, since documents are organized in some semantically meaningful way in these categories, the identification of just a few potentially relevant sub-categories within the hierarchy gives users ready access to a number of related and potentially relevant documents.

While virtually all such topical guides are currently manually constructed, we believe that it is possible to build computation tools which can help to automatically group related documents into sub-categories. Consequently, such tools would help to address the information browsing task without requiring the extensive overhead of manual organization. Moreover, the dynamic applicability of such tools would allow for otherwise unstructured collections of documents to be easily organized in a variety of ways depending on the context (e.g., organizing different query results returned by a retrieval engine).

1.2 System Overview

Gievn the information access challenges outlined above, we now turn our attention to building a system which helps to address some of them. We begin with an overview of a system for *topical* information space navigation that combines both the query-based and directory-based approaches. Our system is named **SONIA**: **S**ervice for **O**rganizing **N**etworked **I**nformation **A**utonomously. This system employs a number

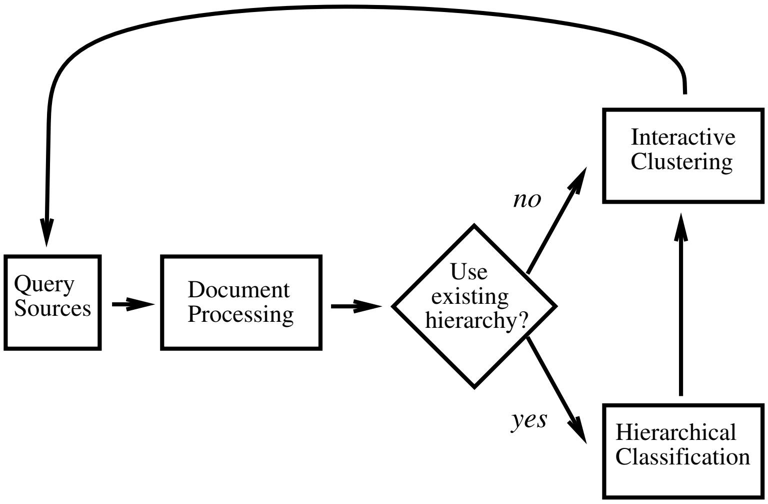

Figure 1.1: Flow chart of user interaction in SONIA.

of machine learning techniques, such as feature selection, clustering, and classification, to create dynamic categorizations based on the full-text of retrieval results. Using this system, users can specify their information needs as queries and browse the results at a topical, rather than document, level.

The technical components of SONIA make use of the formal framework of probability and information theory. As such, these technical components are based on a mathematical foundation that lends itself to a more formal analysis of the assumptions made in these learning models. Since probability theory also provides semantics for such mathematical models, it is also possible to ascertain the degree to which these models capture certain general properties of text domains.

With that said, the best way to get an idea of the functionality that SONIA provides is to consider the user interaction model. Figure 1.1 presents a flow chart of a user's interaction with the system. While we discuss SONIA more fully in Chapter 10 and show examples of actual usage there, the presentation here provides a road-map of the components that comprise the system.

Starting in the standard *ad hoc retrieval* paradigm, SONIA allows users to issue queries to any of a number of available distributed information sources (i.e., networked queriable text collections). It is important to note that SONIA does not maintain its own index of documents and, hence, does not provide its own search engine. As a result, methods for improving the retrieval performance of the system are not a research issue in this work. Rather, we can leverage over 30 years worth of research advances in retrieval [146, 166, 51, 155] as embodied in several on-line search engines [151] by simply allowing SONIA to directly interface with such information sources. Such interfacing is done via the Stanford Digital Libraries InfoBus interoperability protocol [10] and is described in detail in Chapter 10. By relying on existing search engines with very broad coverage, we can both allow SONIA to work with information sources that users may already be familiar with, while at the same time maintain a clean division between the search engine and the subsequent processing that takes place in SONIA. This division makes the system maximally flexible. That is, our system allows for multiple distributed, heterogeneous collections to be queried and also for new information sources to easily be integrated.

Once an initial set of documents is returned in response to a user query, SONIA can then be used to **cluster** these documents into related groupings. Such a clustering allows a user to get a better sense of the topics encompassed in the possibly large number of documents returned in response to a query. We address clustering in more detail in Chapter 6, but it suffices to say here that clustering can be successful at grouping documents based on the similarity of their textual content. The resulting document grouping often resembles a topical category structure automatically imposed on just the set of documents matching a user's query. We believe that this structure enables a user to browse documents much more easily than scanning through a lengthy and unstructured list. Furthermore, re-clustering of the documents in particular groups is possible, thereby creating a hierarchical structure of document groupings. The SONIA user interface also allows a user to rearrange both the structure of this hierarchy as well as the placement of individual documents within it. As a result, users can leverage such a tool to quickly generate a topical directory structure for the documents that are potentially relevant to their information needs (as articulated in their queries).

Notice that having such a hierarchical structure not only helps enable users in the *browsing* task, but creates a partitioning of the information space that allows a user to better organize the information available about a particular topic. Moreover, the automated creation of such a structure helps to bridge the gap between the traditional query-based and directory-based paradigms. Still, as mentioned previously, the manual maintenance of such hierarchical directories can present a formidable challenge. Thus, SONIA provides methods for automatic **classification** of new documents (e.g., the results of subsequent queries, or documents returned by roving Web agents) into such a hierarchy. Note that the topic hierarchy need not be constructed by SONIA, as the classification methods it employs are readily applicable to existing hierarchies that a user may have manually created (e.g., his or her Web page bookmarks, or the hierarchical directory structure of a user's file system). It is important to realize that this ability to classify documents helps to address the *filtering* and *routing* tasks, as these problems can be seen as simply classifying documents into the groupings which are *relevant* and *non-relevant* to the user. In this way, the system can help a user to not only quickly organize a collection of documents, but also maintain this organizational schemes over time by routing new documents to their correct location in the hierarchy.

1.3 Reader's Guide

We now delve deeper into the technical components that comprise SONIA in order to highlight the contributions of this work. While SONIA does embody some elements found in other IR systems, it uses different technologies to realize this functionality. Additionally, the novel machine learning tools developed as part of SONIA are general enough that they can be used on a variety of problems in other domains. While our presentation herein focuses on the evaluation of the technical components of SONIA on primarily textual data, we do present results on other domains in some cases to highlight the full generality of these methods.

The complete interaction model that SONIA provides for users is also novel. We hope to elevate user interaction with information access systems beyond simple one-shot queries and move to addressing users' more persistent information needs. To this end, SONIA extends beyond tools which simply provide document clustering

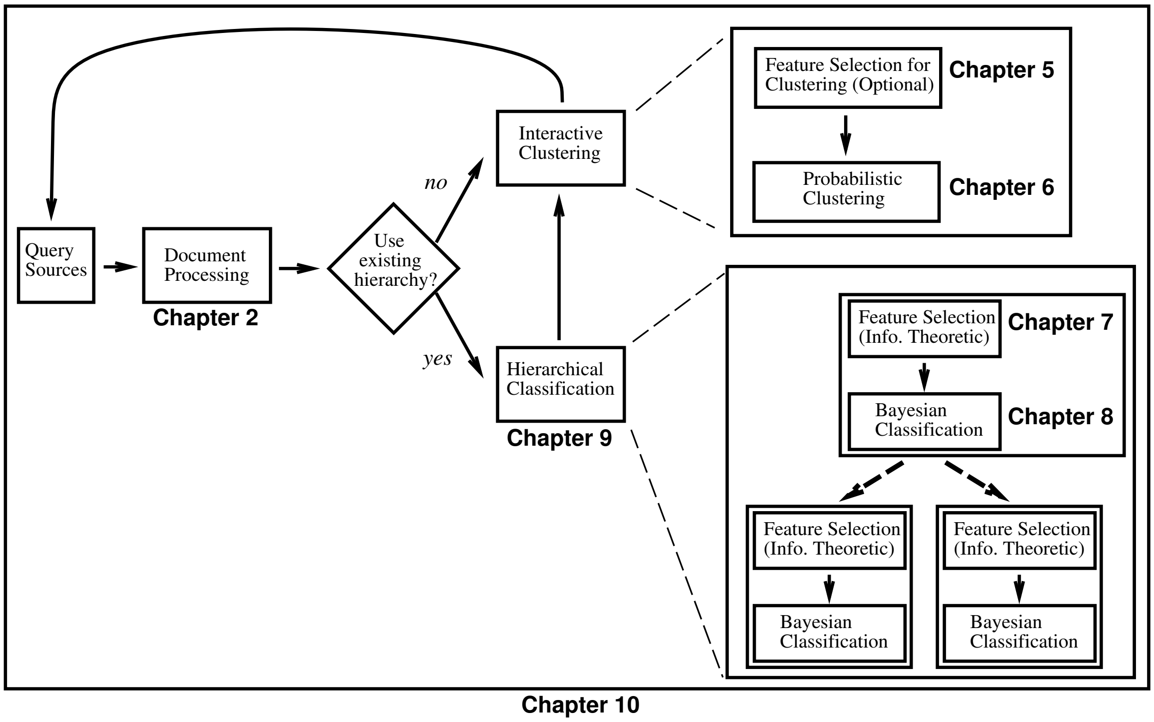

Figure 1.2: Overview of the technical components in SONIA.

capability by providing classification tools and an interactive user interface which allows for such document organizations to be easily expanded and maintained over time.

We now turn our attention to identifying the particular technical components that comprise SONIA, and thereby provide a road-map of the subsequent chapters. Figure 1.2 expands the block diagram of user interaction with the system (seen in Figure 1.1), showing the technical components of SONIA and how they fit together. These modular machine learning components are the foci of Chapters 5 through 9. Here, we briefly outline what these components are and how they are integrated together.

As mentioned previously, we begin with the assumption that various information sources (i.e., search engines) exist which SONIA can communicate with via the Infobus protocol. Still, we can only assume about these sources that they provide SONIA with unstructured lists of pointers (i.e., URLs) to documents which may possibly be relevant to users' queries. This assumption is in accordace with the information generally accessible from commercial search engines on the Web. Thus, to subsequently apply any machine learning technologies to the document lists returned from such a queried information source, it becomes necessary to retrieve the actual texts of these documents and then process this text into a representation that is suitable for our learning algorithms. We discuss such document processing and representation in Chapter 2. Moreover, given the vector space representation for documents that we use, it becomes clear that methods for reducing the high dimensionality of document vectors are needed. This process is called *feature selection* in the Machine Learning community. We describe simple methods for such dimensionality reduction in this chapter and set the groundwork for later discussions of powerful novel feature selection methods.

Having defined our document representation, we then need to lay out the theoretical underpinnings of the probabilistic framework we use in our work. This is the subject of Chapter 3. Therein we describe the formalism of Bayesian networks which is the basis for many of the machine learning algorithms described in subsequent chapters. Moreover, we also provide a brief overview of existing machine learning methods that serve as a baseline for comparison with the novel methods that we present later.

Next, in Chapter 4, we provide a context for the novel contributions in this thesis by presenting previous work in Information Access that is most closely related to the development of SONIA. Specifically, we begin by examining the historical role of probability theory in Information Retrieval. We then show more recent developments in document clustering and classification. We give a survey of this work and point out some of the existing systems that are closest in spirit to SONIA.

Having dispensed with the preliminaries, we dive into the technical components of SONIA as seen in Figure 1.2, beginning first with the topic of clustering. In Chapter 5 we examine the problem of using feature selection as a means for dealing with clustering high-dimensional data with mixture models. We present two novel algorithms for feature selection and give theoretical results showing how these algorithms minimize information loss in the probability distributions being modeled. Moreover, we shows that these methods help focus the clusters recovered via mixture modeling on the most significant probabilistic dependencies in the data. They thereby help to better uncover distinct sub-groups within the data.

Nevertheless, while these methods show promise at identifying features which lead to better resulting mixture models, the absolute performance of the resulting clustering still leaves something to be desired. The poor performance of mixture modeling suggests the need of a better model for document clustering altogether.

Addressing this need is the focus of Chapter 6, which describes joint work with Goldszmidt [63]. Here we construct a model for clustering documents based on a novel probabilistic similarity measure that captures the expected overlap in words between documents. This score prompts the investigation of different methods for estimating the probability of a word appearing in a document. Furthermore, we show that the *cosine coefficient* [131] widely used in information retrieval as a measure of document similarity can be associated with a particular form of probability estimation in our model. We also introduce a specific scoring function that outperforms the cosine coefficient and its extensions in our experiments with document clustering tasks. Furthermore, our experiments indicate that our model outperforms the standard mixture models examined in Chapter 5. We believe that this improvement is due to our model's asymmetrical treatment of positive (word appearance) and negative (word absence) information in the document clustering task. This assymetry does not appear in mixture models, since they weigh information about word appearance and absense equally.

Having found a suitable model for document clustering, we then turn to the topic of classification. In Chapter 7 (which is based on joint work with Koller [94]), we again take up the issue of feature selection, but this time in the context of document classification rather than clustering. We present a method for feature subset selection based on Information Theory [35] that rests directly on our underlying probabilistic framework. Initially, we define a theoretically optimal, but computationally intractable, method for feature subset selection. We show that our goal should be to eliminate a feature if it gives us little or no additional classification information beyond that subsumed by the remaining features. In particular, this will be the case for both irrelevant and redundant features. We then give an efficient algorithm for feature selection which computes an approximation to the optimal feature selection criterion. We also present empirical results on both textual and non-textual data, showing the generality of the algorithm in effectively handling a wide variety of datasets with large numbers of features.

Chapter 8 addresses the issue of learning Bayesian classification models. Realizing that strong probabilistic dependencies often exist between words in documents, we examine methods for learning more expressive models than the Naive Bayesian classifier. These models as refered to as k-dependence Bayesian classifiers and are characterized by the degree of dependence between features in the model. The Naive Bayes algorithm is shown be the most restrictive model according to this characterization, while the learning of full Bayesian networks is at the most general extreme. We present an induction algorithm that allows for models characterized by various complexity to be quickly learned. Analyzing the modeling assumptions made as one creates richer models, we show empirical evidence of the tension between model expressivity (i.e., bias) and the variance associated with fitting the parameters of a very expressive model when limited data is available. This phenomenon is especially acute in text domains, but we also show applications of our algorithm to a number of other domains with different properties.

Again, describing work done jointly with Koller [95], Chapter 9 addresses the goal of automatically classifying new documents into *hierarchical* classification taxonomies. We note that, up to now, existing classification schemes have ignored the hierarchical relationship of topics; most simply treat the topics as separate classes. As a result, such methods can be inadequate for text classification where there is a large number of classes and a huge number of relevant features may be needed to distinguish among them. Rather, we propose an approach that utilizes the hierarchical topic structure to decompose the classification task into a set of simpler problems, one at each node in the classification tree. As we show, each of these smaller problems can be solved accurately by using the feature selection algorithm from Chapter 7 to focus on only a very small set of features, those relevant to the task at hand. This set of relevant features varies widely throughout the hierarchy, so that, while the overall relevant feature set may be large, each classifier only examines a small subset. The use of reduced feature sets allows us to utilize more complex (probabilistic) models, such as those presented in Chapter 8, without encountering many of the computational and robustness difficulties previously outlined.

Having described all the technical components in SONIA, in Chapter 10 we can finally give a detailed description of the whole system in operation. First, we describe how the system is integrated in the Stanford Digital Libraries testbed, and how it makes use of the InfoBus protocol to communicate with a large number of distributed, heterogeneous information sources. Then, we provide several anecdotal examples of user interaction with the system, giving evidence of SONIA's efficacy in helping users to organize and browse the results of queries. We also show how SONIA can be used in a very different application to help a user create and maintain a hierarchical organization of documents in their local file directory structure. As part of these examples, we present the interactive user interface of the system and point out how it integrates document content browsing with tools for document organization.

Finally, in Chapter 11, we provide some brief concluding remarks on the general lessons learned from this work. Specifically, we consider how controlling the variance of induced model parameters seems to be a critical issue in the success of applying learning methods to problems in text domains. We conclude by outlining other potential applications of our work and presenting promising directions for future research.

Chapter 2

Document Representation

In order to apply the machine learning tools described in subsequent chapters to text documents, we need to represent documents in a way amenable to probabilistic reasoning. Initially, we may be tempted to consider various natural language document representations from the computational linguistics community [21] (e.g., parse trees). Such representations, however, are generally used to address problems such as the part-of-speech tagging and are not as directly useful for the types of clustering and classification problems we consider.

Rather, we choose to employ a vector space representation of documents, described in detail in Section 2.1. In this representation, documents are cast as vectors in a very high-dimensional space. Since probabilistic models can be computationally expensive to apply and may suffer from lack of robustness in spaces with such high dimensionality, we examine some simple initial methods for dimensionality reduction in Section 2.2. Figure 2.1 shows the progression of stages involved as text documents are processed into a vector representation with reduced dimensionality.

2.1 Defining a Vector Space

We use the *vector space* [147] representation for documents commonly employed for many information access problems. In this representation, each document is characterized by a Boolean or numerical vector. These vectors are embedded in a space

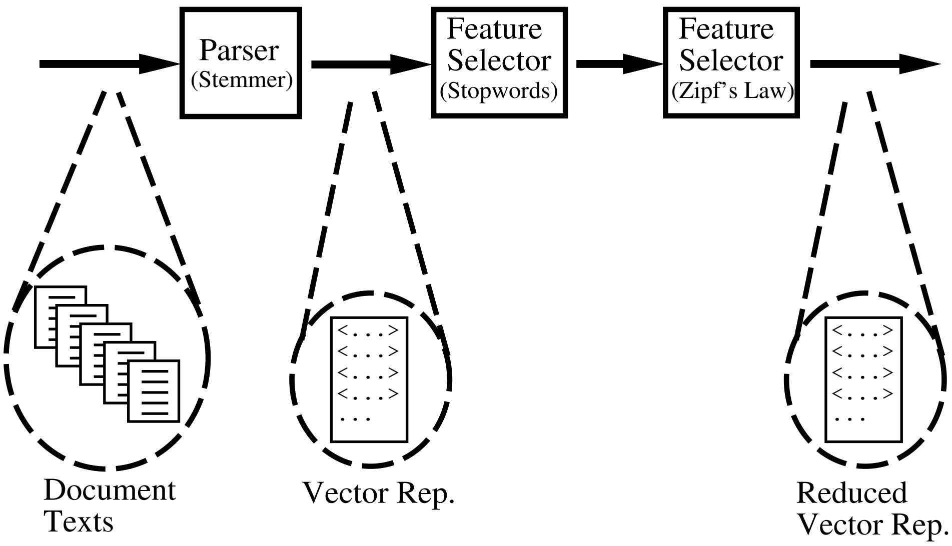

Figure 2.1: Initial document processing and representation phases.

in which each dimension corresponds to a distinct *term* in the corpus of documents being characterized. A given document vector has in each component a numerical value denoting some function f of how often the term corresponding to that dimension appears in the document. By varying the function f, we can produce alternative term "weightings" [145]. We explore some of standard weighting functions below, and give examples of vector representations of documents using these schemes.

2.1.1 Defining "Terms"

Although we have mentioned that the dimensions of our vector space correspond to distinct terms in the corpus of documents being represented, we have not defined what is to be considered a "term." The most common definition of a term (in English, as well as most other languages that use the Roman alphabet), and the one that we use in all subsequent chapters, is that a term is a sequence of alpha-numeric characters which is delimited by white space (spaces, tabs or newline characters) or punctuation marks (such as a period or a comma). Moreover, all uppercase letters in a document are converted to lowercase, so effectively capitalization is ignored.

Consider the two short sample documents below (the first of which contains a quotation from Negroponte [120]).

Computing is not about computers any more. It is about living.

Sample document 1.

To live is to compute!

Sample document 2.

Table 2.1 shows the results of parsing these two documents into single-word terms, and then representing them as vectors with simple term frequencies (i.e., term counts) in each component. Such a representation is sometimes also referred to as a *bag of words* [117], since the relative position of terms in the document, and hence the language structure, is not captured in the resulting vectors.

Term	Vector for document 1	Vector for document 2
about	2	0
any	1	0
compute	0	1
computers	1	0
computing	1	0
is	2	1
it	1	0
live	0	1
living	1	0
more	1	0
not	1	0
to	0	2

Table 2.1: A simple vector representation of the sample documents.

Word Stemming

In some cases, rather than defining terms to be the distinct words in the corpus, word *stemming* is used to reduce words to some root form. Thus, the terms that define the dimensions of the vector space are not actual words, but word stems. For example, the words "computer", "computers", and "computing" would all be reduced to the word stem "comput". Porter [128] has developed a commonly used algorithm for word stemming and this algorithm has been incorporated as an optional feature in SONIA's document parsing module.

Comput i not about comput ani more. It i about live.

Stemmed version of sample document 1.

To live i to comput!

Stemmed version of sample document 2.

Above we show what the two sample sample documents presented earlier would look like if their contents were stemmed using Porter's stemming algorithm. Table 2.2 shows the vector representation of the stemmed version of the documents. While it is clear that, in some cases, stemming may be useful to help conflate similar terms (such as the stem "comput"), in other cases the results of stemming are counter-intuitive (such as stemming "is" to "i"). Frakes [50] provides an overview of studies comparing various stemming methods to unstemmed representations for the retrieval task and shows that in many cases both representations perform roughly equally. Consequently, in the remainder of this work we do not make use of stemming, since we believe it will not have a significant impact on performance, but would make qualitative examples more difficult to understand.

Stem	Vector for document 1	Vector for document 2
about	2	0
any	1	0
comput	2	1
i	2	1
it	1	0
live	1	1
more	1	0
not	1	0
to	0	2

Table 2.2: A vector representation of the stemmed version of the sample documents.

Multi-word terms

Some researchers have defined the dimensions of a vector space to include multi-word phrases. Examples of such multi-word terms include phrases such as "President Clinton" and "personal computer." Such multi-word terms can be produced as a result of simply looking for frequently appearing sequences of words in the documents [29], applying natural language processing to detect meaningful phrases [55] or hand-engineering specific phrases for particular tasks [140, 156].

As with stemming, previous results with using multi-word terms are mixed. Some researchers report that using such terms can help improve accuracy for classification tasks [29], whereas others have found them to be no more effective than simple single-word terms [48]. Some of this discrepancy can be accounted for by the expressivity of the models used for learning. For example, models that do not capture co-occurrence information between words may stand to benefit from multi-word features which explicitly express such dependencies. On the other hand, models capable of learning word co-occurrence information may not gain anything from the inclusion of multi-word phrases and may actually be hindered by having to estimate additional parameters for such terms. Since, in Chapter 8, we employ learning models capable of capturing probabilistic dependencies between words, we only consider vector spaces defined by single-word terms in our work.

2.1.2 Frequency-based Vectors

As mentioned previously, in a vector space representation of documents, various functions may be applied to the frequency of term occurrences in documents in order to produce "weighted" document vectors. More formally, let $\xi(t_i, d)$ denote the number of occurrences of term t_i in document d. We may then apply some function f to $\xi(t_i, d)$ to produce the value for the i-th component of the vector for document d. For the vectors in Table 2.1, for example, we simply used the identity function $f(\alpha) = \alpha$ applied to the term counts.

Other common functions applied to term frequencies include:

$$f(\alpha) = \log(\alpha + 1) \ , \tag{2.1}$$

which was defined by Robertson and Sparck Jones [135] and used successfully for retrieval;

$$f(\alpha) = \sqrt{\alpha} \ , \tag{2.2}$$

which was used in the Scatter/Gather system [39] for document clustering and was found to outperform Eq. 2.1 for that task;

$$f(\alpha) = \frac{\alpha}{\alpha + \mathrm{Const}} \ , \tag{2.3}$$

which was proposed by Robertson and Walker [136], who found this general form to be useful for document retrieval (using various instantiations of the constant value).

Perhaps the most well-known function applied to document term frequencies is TFIDF weighting [145]. In this scheme, not only are the term frequencies (TF) in each document used as part of the weighting function, but so is the inverse document frequency (IDF) of each term in the entire collection. More formally, IDF is usually defined as

$$\mathrm{IDF}(t) = \log(\frac{N}{n_t}) \ , \tag{2.4}$$

where N is the total number of documents in the collection and n_t is the number of documents in which term t appears at least once.

The TFIDF weight for a term t in a document d is the product of the term frequency and the inverse documents frequency for that term, yielding:

$$\mathrm{TFIDF}(t, d) = \xi(t, d) \cdot \mathrm{IDF}(t) \ . \tag{2.5}$$

Although TFIDF weighting has been used in the past primarily for retrieval, connections between this weighting scheme and probabilistic classification using the Naive Bayes algorithm have been recently explored by Joachims [81].

2.1.3 Boolean Vectors

Alternatively, we may consider using a simple Boolean representation of documents, in which we simply record whether or not a given term appears in a document. In this case, we have:

$$f(\alpha) = \begin{cases} 1 & \alpha \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

Table 2.3 shows the sample documents from Section 2.1.1 cast as Boolean vectors. Note that most rule-based methods [7, 28] are essentially using an underlying Boolean model (even if it is not explicitly defined as such), as the antecedents of the classification rules they produce are only considering word presence and absence in documents.

We make use of the Boolean vector representation in our probabilistic classification models. While at first, it may seem that we are creating a disadvantage for ourselves by not considering the frequency of word appearances, this is not necessarily the case. In actuality, it can be problematic to make full use of word frequency information in probabilistic classification. There are two standard approaches for incorporating such information, which are described below.

The first approach involves using a non-parametric model of term frequencies in documents (i.e., using a multinomial as opposed to binomial distribution per term). In this case, each distinct number of times that a word appears in a document is

Term	Vector for document 1	Vector for document 2
about	1	0
any	1	0
compute	0	1
computers	1	0
computing	1	0
is	1	1
it	1	0
live	0	1
living	1	0
more	1	0
not	1	0
to	0	1

Table 2.3: A Boolean vector representation of the sample documents.

represented by a distinct value of a multinomial distribution. Clearly, we quickly run into problems of estimating model parameters from limited data as we need to estimate a parameter for *each* number of times that a word can appear in a document. Still, in preliminary experiments [48], we tried to control for this blow up in the parameter space by considering limited frequency information. To this end, we tried using a rough discretization of the feature counts into three-valued attributes corresponding to the events that a word did not appear in a document, appeared only once in the document, or appeared two or more times in the document. The results of these experiments showed virtually no difference between the Boolean and discretized frequency representations.

The second approach to incorporating word frequency information into probabilistic classification models is by using a parametric distribution (e.g., bounded Gaussian or Poisson distribution) to capture the probability of words appearing different numbers of times in documents. Here, one must commit to a particular parametric form for the distribution of words in documents, even though there may be no compelling reason to favor any particular distribution (although this may be an interesting venue for future work). Our initial investigations conducted along these lines using bounded Gaussian distributions versus a simple Boolean representation have not shown this to be a promising venue. Further evidence for the adequacy of the Boolean representation is reported by Yang and Chute [174], who, in comparing Boolean and frequency-based representations in the context of instance-based classification, did not observe much difference between these representations.

We do point out, however, that in Chapter 6, we make use of word frequency information in documents. Still, it is important to note that there we are considering an entirely new model for document clustering (based on repeated draws from a single multinomial distribution) and are thus not working with the same models that we use subsequently for classification.

2.2 Controlling Dimensionality

In using a vector space representation for documents, it becomes clear that the resulting dimensionality of the space will be enormous, since the number of dimensions is determined by the number of distinct terms in the corpus. For example, feature spaces on the order of 10^3 to 10^5 are not uncommon for even reasonably small collections of documents. The problem of high dimensionality is further exacerbated in very heterogeneous environments, such as the World Wide Web.

Thus, methods for controlling the dimensionality of the vector space are needed. Here, we show how it is possible to use a few simple observations to reduce the size of the feature space significantly. These dimensionality controls, however, are crude and are not directly based on clustering and classification problems which we hope to solve in the resulting vector space. Consequently, we also present more sophisticated, task-directed feature selection methods in Chapters 5 and 7, where we discuss feature selection in the context of clustering and classification, respectively.

a	been	do
able	before	does
about	below	during
after	best	each
again	but	else
all	by	enough
almost	came	ever
also	can	except
am	cannot	few
and	clearly	for
are	come	former
as	consider	from
at	could	get
be	despite	goes
because	did	going

Table 2.4: Sample list of stop words.

2.2.1 Eliminating Stop Words

Initially, we make an observation that is common throughout the IR literature [166]: there exist many words in English which have little inherent topical content. These are words such as prepositions, conjunctions and pronouns that are used to provide structure in language rather than content. Such words are commonly used in documents regardless of topic, and thus have no topical specificity. As a result, we can eliminate such words (and the dimensions corresponding to them) from our document vectors, as they will be little use when clustering or classifying documents. Such words are commonly referred to a *stop* words, and their elimination from documents is common in IR. A list of exemplary stop words is given in Table 2.4.

Currently, in SONIA we eliminate terms found on a stop word list of approximately 570 common English words and an additional 100 commonly used words on the Web (e.g., "click" and "page"). While this serves to eliminate several hundred dimensions from our vector space, given the high-dimensionality of text, additional term elimination is necessary.

2.2.2 Zipf's Law

To further reduce the dimensionality of the document vectors, we make use of another well known phenomenon: many words in a corpus appear very infrequently. If our goal is to identify similarities and differences among an entire collection of documents, then words which only appear, say, once or twice (or generally infrequently) in the collection will have little resolving power between documents [166].

The justification for the elimination of such infrequent terms lies in an observation about the frequency of word appearances in corpora made by Zipf over 50 years ago [177]. Since that time, this observation has been named "Zipf's Law," although it is not actually a law, but merely an empirical and approximate mathematical phenomenon.

To describe Zipf's Law more formally, let us denote the total frequency of a term t in a corpus D by ζ_t. That is, $\zeta_t = \sum_{d \in D} \xi(t,d)$. Then, we sort all terms in descending order according to ζ and give each term t a rank r_t based on its placement in the sorted list. Zipf's Law states that

$$r_t \cdot \zeta_t \approx K \ , \tag{2.7}$$

where K is a constant. This relationship is shown graphically in Figure 2.2. In English text, it has been observed across a variety of collections that $K \approx \frac{N}{10}$, where N is the total number of words in the corpus [18].

Following Callan's treatment [18], let us rewrite Zipf's Law as $r_t \approx \frac{K}{\zeta_t}$. Now, consider term a which is the lowest ranked term with some frequency β, and term b which is the lowest ranked term with frequency $\beta+1$. We can obtain the approximate ranks of these terms as $r_a \approx \frac{K}{\beta}$ and $r_b \approx \frac{K}{\beta+1}$, respectively. Subtracting these two ranks, in turn, gives us an approximation to the number of distinct terms that have frequency equal to β. That is:

$$\begin{aligned} r_a - r_b &\approx \frac{K}{\beta} - \frac{K}{\beta+1} & (2.8) \\ &= \frac{K}{\beta \cdot (\beta+1)} \ . & (2.9) \end{aligned}$$

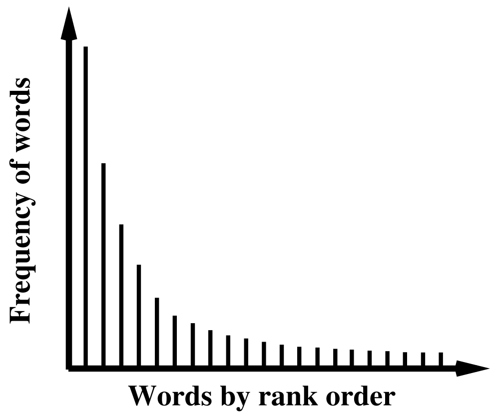

Figure 2.2: Histogram of word frequencies showing Zipf's Law.

Now, estimate the number of distinct terms in the collection by computing the rank of the highest rank term in a collection. Generally, this will be a term which only appears once in the collection, giving us:

$$r_{max} \approx \frac{K}{1} = K \ . \tag{2.10}$$

If we now consider the fraction of unique terms that appear β times in a collection, we can simply divide Eq. 2.9 by Eq. 2.10 to obtain $\frac{1}{\beta \cdot (\beta+1)}$. Thus, Zipf's Law shows us that a significant fraction of the distinct terms in a collection are made up by those terms that appear the most infrequently. For example, the number of terms that appear only once in the corpus will account for approximately $\frac{1}{2}$ of the total unique terms. Even recalling that this "law" is really only a loose approximation, it still provides compelling evidence that we can eliminate a significant number of terms (i.e., dimensions in the corresponding vector space) by simply eliminating the terms with the lowest frequency of occurrence in the corpus. And as noted earlier, such terms will be of little use for clustering or classification.

Consequently, SONIA incorporates a module for eliminating infrequent terms from a collection, and such computation can be done extremely quickly. While there is still a question of what frequency threshold to use when doing such feature elimination, in practice it often depends on making a judgment as to how many features we believe are realistic to use with the algorithms that are applied after this initial processing. In the experiments described in later chapters, we regularly use such Zipf's Law-based feature selection to eliminate all terms that appear fewer than 10 times in a given document collection. This value was heuristically chosen after some initial experiments, as it generally reduces the feature space to on the order of a few thousand features, which is a feasible size to use as input to our more sophisticated feature selection algorithms. Moreover, the threshold is still conservative enough that it does not appear to eliminate features which could have a large impact on overall performance.

Chapter 3

Probabilistic Framework

Given that we now have a representation of documents as Boolean vectors (and have eliminated stop words and terms occurring fewer than 10 times in the collection), we now present a theoretical framework for developing machine learning tools to apply to such document vectors. We begin in Section 3.1 by giving an overview of Bayesian networks–a graphical formalism for reasoning about probability distributions and the main framework we use in the technical components of SONIA. In Section 3.2, we then give a brief overview of some existing machine learning algorithms that are used later in this work for means of comparison.

3.1 Bayesian Networks

In order to reason about documents, we use the calculus of probability theory. At the heart of the probabilistic framework is the idea that our model of the world (in this case, a collection of documents) is represented as a probability distribution over the space of possible states of the world. Typically, a state of the world is described via some set of random variables, so that each such state is an assignment of values to these variables.

A Bayesian network [125] provides a compact description of a complex probability distribution over a large number of random variables. It uses a directed acyclic graph (DAG) to encode *conditional independence* assumptions about the domain. Each variable (also referred to as a *feature*) X_i is represented as a node in the network. An arc between two nodes denotes the existence of a direct probabilistic dependency between the two variables. Hence, the *lack* of an arc between two nodes implies that no *direct* probabilistic influence exists between those variables. Essentially, the structure of the network denotes the assumption that each node X_i in the network is conditionally independent of its non-descendants given its parents $\Pi(X_i)$.

Definition 1 *Two sets of variables are said to be* conditionally independent *given some set of variables* $\boldsymbol{X}$ *if, for any assignment of values* $\boldsymbol{a}$, $\boldsymbol{b}$, *and* $\boldsymbol{x}$ *to the variables* $\boldsymbol{A}$, $\boldsymbol{B}$, *and* $\boldsymbol{X}$ *respectively,* $P(\boldsymbol{A} = \boldsymbol{a} \mid \boldsymbol{X} = \boldsymbol{x}, \boldsymbol{B} = \boldsymbol{b}) = P(\boldsymbol{A} = \boldsymbol{a} \mid \boldsymbol{X} = \boldsymbol{x})$. *That is,* $\boldsymbol{B}$ *gives us no information about* $\boldsymbol{A}$ *beyond what is already in* $\boldsymbol{X}$. *(See [125] for more details).*

Independence assumptions allow the distribution to be described as a product of small *local interaction models*. To describe a probability distribution satisfying these assumptions, we associate with each node X_i in the network a *conditional probability table* which specifies the distribution over the possible values of X_i given any possible assignment of values to its parents $\Pi(X_i)$. If X_i has no parents, it simply contains a prior probability distribution over X_i's values. The network structure and the associated parameters uniquely define a probability distribution over the variables in the network.

In the case of documents, for example, we can define one random binary variable (i.e., node in the network) to correspond to each distinct term in the vector space. A document then is an assignment to these variables reflecting which terms appear in the document vector and which do not. In other words, the vector representing a document defines a unique assignment to all of the feature nodes in the Bayesian network.

To make this description more concrete, consider the simple example Bayesian network in Figure 3.1. This network represents a domain of six terms (i.e., vector space of six dimensions) and encodes some of the probabilistic dependencies that we might expect to see between words in a document collection. If we represent the probability distribution over these six binary variables *without* considering the conditional

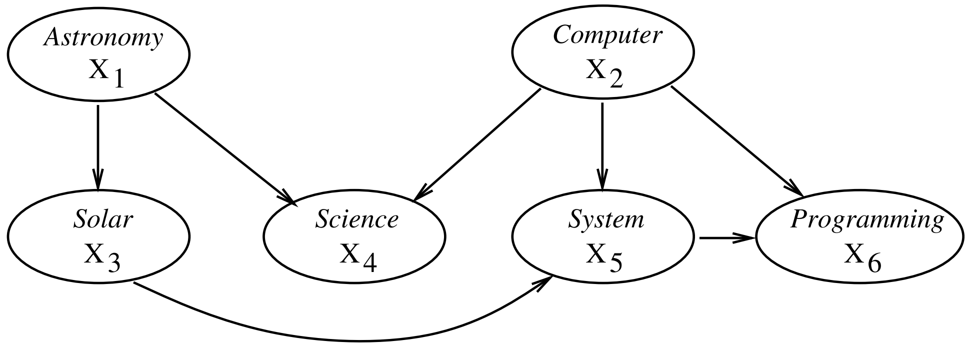

Figure 3.1: A simple example of a Bayesian network.

independencies encoded in the network structure, it would require a joint probability table containing $2^6 - 1 = 63$ parameters: one parameter for each possible assignment to the six variables, except we subtract one since the distribution is constrained to sum to one. Alternatively, using the independencies encoded in the network, we can decompose the distribution over these six variables as

$$\begin{aligned}&\mathrm{P}(X_1, X_2, X_3, X_4, X_5, X_6) = \\ &\quad \mathrm{P}(X_1) \cdot \mathrm{P}(X_2) \cdot \mathrm{P}(X_3 \mid X_1) \cdot \mathrm{P}(X_4 \mid X_1, X_2) \cdot \mathrm{P}(X_5 \mid X_2, X_3) \cdot \mathrm{P}(X_6 \mid X_2, X_5)\end{aligned}$$

Note, the distribution factored in this form only requires $1 + 1 + 2 + 4 + 4 + 4 = 16$ parameters to specify—far fewer than the case where we assumed no independencies existed between the variables.

For text modeling problems, we will generally have so many features to model that it is simply intractable to build a probabilistic model without incorporating some independence assumptions. We believe that Bayesian networks provide an excellent means for controlling the computational complexity of our models through the ability to easily assert such assumptions. The question then becomes one of *how much* independence to assume and we address this issue in a variety of contexts throughout this work.

Another benefit of Bayesian networks is that they allow for domain knowledge to be easily incorporated. For example, if we had a priori knowledge of word dependencies in a document collection, we could directly encode them in the network and, hence, not require a learning algorithm to have to discover these dependencies by itself. While we do not actually make use of this feature in our work (since we are focusing on entirely automated learning methods), we point out this fact to show that the formalism we harness lends itself to such extensions. Similarly, since Bayesian networks are graphical in nature, it is far easier for a human expert to validate (and possibly correct) learned models than in the case of other learning frameworks, such as neural networks, which are far more opaque.

Finally, by operating in the realm of probability theory, Bayesian networks have understandable semantics. This is an important factor in helping researchers working with such tools to gain understanding about a novel domain by examining (automatically) constructed models of it. Moreover, we will explain in Chapter 4 how probability has been previously advocated as a formalism for advancing the state of the art in the information retrieval task. We hope to do the same here for the tasks of clustering and classifying documents.

While the formalism of Bayesian networks has existed for over a decade, it has only been in the past few years that this framework has been employed with the goal of automatically learning the graphical structure of such a network from a set of data [31, 17, 72]. Presently, we describe several learning methods, most of which are based directly on Bayesian networks, that are aimed at the specific tasks of clustering and classification. We then apply these methods in subsequent chapters on text domains.

3.2 Machine Learning Overview

3.2.1 Classification

Document classification involves assigning documents to one of some set of *pre-defined* categories. In the classification task we are given a training set of data with pre-assigned class labels. From this data, we then learn a model that can be used to classify new data into one of the existing classes.

Formalizing this notion using notation similar to our treatment of clustering, we being with a set D_{train} of m training documents, denoted $d_1, \ldots, d_m$. Each such

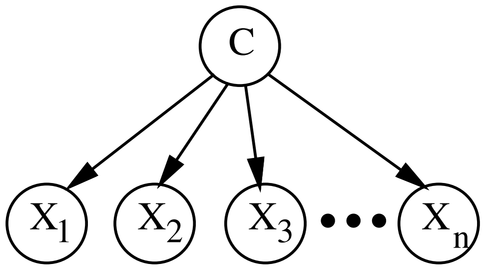

Figure 3.2: The network structure for Naive Bayes.

document also has associated with it a class label, denoted by the variable C. Learning a classifier, also known as *supervised* machine learning (since we begin with a set of pre-labeled data), involves inducing a model from the training data that we believe will be effective at predicting the class label C in new data for which we do not know the class. This new data is often referred to as the testing set D_{test} in experimental evaluations.

We now turn to the question of how to induce a classification model from a pre-labeled set of data. First, we provide a brief introduction to Bayesian network classifiers—a topic we treat more fully in Chapter 8. We also give an overview of decision tree induction, as this is a commonly used learning method that we employ in the comparative experiments presented in Chapter 7.

Bayesian Network Classifiers

A Bayesian network classifier is simply a Bayesian network applied to a classification problem. It contains a node C for the class variable and a node X_i for each of the domain features (i.e., words, in the case of documents). Given a specific instance vector $\boldsymbol{x}$ (an assignment of values $x_1, x_2, \ldots, x_n$ to the feature variables), the Bayesian network allows us to compute the probability $\mathrm{P}(C = c_k \mid \boldsymbol{X} = \boldsymbol{x})$ for each possible class c_k. If the true distribution $\mathrm{P}(C \mid \boldsymbol{X})$ were known, we could achieve *Bayes Optimal* classification by simply selecting the class c_k for which this probability is maximized. Unfortunately, the true distribution is not available and, hence, must be approximated (i.e., learned) from the training data D_{train}.

While it is possible to use any Bayesian network over these variables as a Bayesian

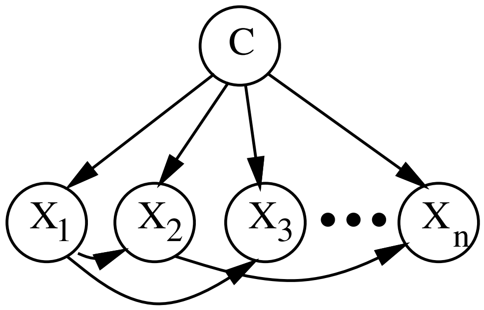

Figure 3.3: A Bayesian classifier allowing limited dependencies between the features.

classifier [153], empirical evidence [53] suggests that networks where the feature variables are all directly connected to the class variable are better at the classification task. The simplest and earliest such classifier is the Naive Bayesian classifier [64]. This classifier, which significantly predates the development of Bayesian networks, is still widely employed today. The Naive Bayesian classifier makes the simplifying, but very restrictive, assumption that domain features are conditionally independent of one another, given the class variable. In other words: $\mathrm{P}(\boldsymbol{X} \mid C) = \prod_i \mathrm{P}(X_i \mid C)$. This assumption corresponds to the Bayesian network structure shown in Figure 3.2.

The assumption that all features are conditionally independent given the class variable is clearly unrealistic in text, as well as many other domains. Still, the Naive Bayesian classifier has been shown to be surprisingly effective on such problems [117]. Domingos and Pazzani [43] present an argument for the efficacy of Naive Bayes for classification, but concede that it often does not produce accurate estimates of the actual probability for class membership.

In an effort to build more accurate classification models, several approaches have been proposed for augmenting the Naive Bayesian classifier with limited interactions between the feature variables, i.e., where we allow each node to have some parents beyond the class variable (as illustrated in Figure 3.3). Unfortunately, the problem of inducing an optimal Bayesian classifier is NP-hard even if we restrict each node to have at most two additional parents in addition to the class variable [25]. Thus, any optimal algorithm for constructing such a classifier would be exponential in the number of features in the worst case.

Several solutions have been proposed to this problem. One of the most well-known is the Tree Augmented Naive-Bayes (TAN) algorithm of Friedman and Goldszmidt [53] which restricts each node to have at most one additional parent beyond the class variable. In this case, an optimal classifier can be found in quadratic time (in the number of features). More formally, TAN constructs an optimal approximation (in terms of the Kullback-Liebler divergence [100]) to $\mathrm{P}(\boldsymbol{X} \mid C)$ which has the general form $\prod_i \mathrm{P}(X_i \mid X_{i'}, C)$, where $X_{i'}$ is the single parent (not counting the class node C) of feature X_i in the underlying Bayesian network.

In our work, we also address the problem of learning Bayesian network classifiers with an alternative algorithm named KDB (k-Dependence Bayes) [139]. We save the description of this algorithm for Chapter 8.

Decision Trees

While decision trees are not based on the formalism of Bayesian networks, they are one of the most popular machine learning techniques currently used. To make the use of decision trees in the context of text classification more concrete, consider the decision tree presented in Figure 3.4. This illustrative example shows what a typical decision tree might look like for a classifier that tries to distinguish documents as being about the topic "Electronic Commerce" or not. Each node in the tree tests whether or not a given word appears in the document, and the respective branch is then followed down the tree until a leaf node, or final classification, is reached.

Several methods have been proposed for inducing decision trees, including the CART algorithm of Breiman *et al* [15], Quinlan's ID3 algorithm [129], and its more recent incarnation as C4.5 [130]. Bayesian methods for constructing decision trees have also been proposed by Buntine [16]. Our treatment here focuses on the C4.5 algorithm since it is the most widely used tree induction method in the machine learning community. We also use this algorithm directly in some of our subsequent experiments.

C4.5 induces a decision tree in a greedy divide and conquer fashion. As the tree is being constructed, the choice of which feature to split on at a given node is made

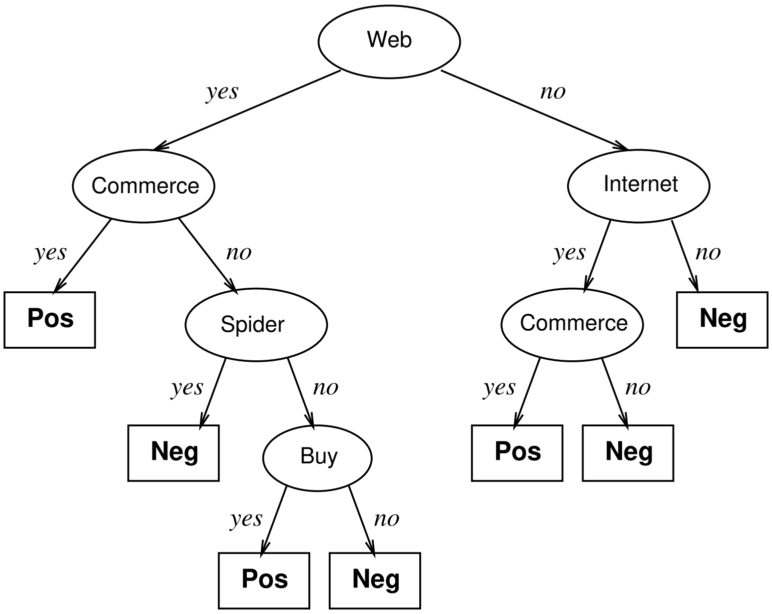

Figure 3.4: An example of a decision tree for the topic "Electronic Commerce."

by selecting the feature which maximizes the *gain ratio* criterion. More formally,

$$\text{gain ratio}(X_i) = \frac{\text{gain}(X_i)}{\text{split info}(X_i)} \tag{3.1}$$

The numerator in Eq. 3.1 is defined as follows:

$$\begin{aligned} \text{gain}(X_i) &= -\sum_c \mathrm{P}(c) \log \mathrm{P}(c) + \sum_{c,x_i} \mathrm{P}(c, x_i) \log \mathrm{P}(c \mid x_i) && (3.2) \\ &= H(C) - H(C \mid X_i) && (3.3) \\ &= MI(C; X_i) && (3.4) \end{aligned}$$

As seen in the derivation above, the *gain* function is simply a measure of the reduction in entropy H of the class variable C after the value for the feature X_i is observed. This is exactly equivalent to the information theoretic notion of mutual information between the class C and the feature X_i, which we denote $MI(C; X_i)$.

The denominator in Eq. 3.1 is given by

$$\text{split info}(X_i) = -\sum_{x_i} \mathrm{P}(x_i) \log \mathrm{P}(x_i) \tag{3.5}$$

$$= H(X_i) \tag{3.6}$$

This function simply represents the entropy in the variable X_i, which can alternatively by seen as the reduction in entropy realized by selecting this variable to split on.

Armed with Eqs. 3.3 and 3.6, it becomes clear that *gain ratio* is simply a measure of the proportion of information relevant to the class variable C derived from splitting on a given feature X_i. Quinlan explains that the *split info* function was introduced in Eq. 3.1 as a means of preventing the *gain* function from overly favoring attributes with many values. He also reports that using *gain ratio* as a splitting criterion often empirically outperforms the use of *gain* alone, even on entirely binary data. This is important to note since we use binary vectors to represent text documents.

3.2.2 Clustering

In contrast to classification, document clustering involves discovering a set of categories to which documents should be assigned. Thus, rather than trying to simply assign documents to pre-defined categories using a labeled set of data for training, clustering algorithms are required to discover distinct categories using an *unlabeled* set of data. Documents in the dataset are then assigned (often as a by-product of the clustering process) to these newly discovered categories.

To define more formally the notion of document clustering, consider a set D of m documents, denoted $d_1, \ldots, d_m$. We want to partition D into K clusters, $c_1, \ldots, c_K$. Note that this partitioning may either be a *hard* partition, where each document is assigned to be a member of only one clustering, or a *soft* partition in which each document is probabilistically assigned to multiple clusters. Often, hard partitions can simply be thought of as the special case where a document is assigned to the single cluster for which it has the greatest probability.

The clustering task thus has two important components. The first is determining how many clusters to partition the data into (i.e., choosing K). The second is how to assign each document to these respective clusters. From a different perspective, we can also cast this second sub-problem as one of determining the probability that a document may have been generated by the *source distribution* represented by a cluster.

In this work, we focus almost entirely on the second sub-problem of assigning documents to clusters. The reasons for this particular focus are two-fold. First, since we will embed our clustering technology within an interactive user interface, we will leave the question of how many clusters to generate to the user. This gives the user much more power to select the desired level of structural granularity when browsing a document collection. Moreover, such flexibility also allows the user to interactively cluster a collection of documents into a number of different possible partitionings and choose the one which best aligns with his or her organizational needs. Second, we have found from preliminary experiments (some of which are described in Chapter 5) that several of the well-founded methods for trying to automatically determine the number of clusters in a data set generally break down in practice on text data containing even a few hundred terms. While the automatic determination of the number of clusters in a data set is certainly a very important area of on-going research, it is beyond the scope of our current work.

AutoClass

We give a brief overview of the AutoClass system [22, 23], a Bayesian method for clustering whose underlying model is straightforward to characterize as a Bayesian network. AutoClass is a method for *mixture modeling* in which a set of data is modeled as being generated from a mixture of several distributions (i.e., the mixture components). Each of these component distributions can, in turn, be viewed as the primary generator for some subset of the data. Each such subset can then be interpreted as a cluster, since all data instances in that cluster are generated from the same mixture component and thus must possess some internal consistencies.

More formally, given the data set D, AutoClass attempts to construct a mixture model M which maximizes the log-likelihood of observing D, $\log \mathrm{P}(D \mid M)$. Performing this optimization is intractable unless several underlying assumptions are made. Foremost, AutoClass assumes that the data points, or in our case document vectors, are independently distributed, yielding

$$\begin{aligned} \log \mathrm{P}(D \mid M) &= \log \prod_{i=1}^{m} \mathrm{P}(d_i \mid M) && (3.7) \\ &= \sum_{i=1}^{m} \log \mathrm{P}(d_i \mid M) \ . && (3.8) \end{aligned}$$

Now, to construct a mixture model, we must expand $\mathrm{P}(d_i \mid M)$ as a sum over all the mixture components, or clusters, $C = c_k$ as follows

$$\mathrm{P}(d_i \mid M) = \sum_{k=1}^{K} \mathrm{P}(d_i \mid c_k, M) \cdot \mathrm{P}(c_k \mid M) \ . \quad (3.9)$$

This equation now mirrors the standard mixture model equation used in unsupervised learning [46].

At this point, it is important to recall that a document d really denotes an n-dimensional vector of features $X_1, \ldots, X_n$. Consequently, $\mathrm{P}(d \mid c_k, M)$ in Eq. 3.9 expands to $\mathrm{P}(X_1, \ldots, X_n \mid c_k, M)$, which is intractable to compute without further assumptions.

AutoClass makes the strong assumption that each feature is conditionally independent of every other feature, given the mixture component C. This assumption of independence is the same as that made in the Naive Bayesian classifier described above. Recall that this assumption can be expressed mathematically as

$$\mathrm{P}(X_1, \ldots, X_n \mid C, M) = \prod_{i=1}^{n} \mathrm{P}(X_i \mid C, M) \ . \quad (3.10)$$

Similar to Naive Bayes, we can represent this assumption of independence, as well as the general form of the probabilistic mixture model used in AutoClass, in Figure 3.2. In AutoClass, however, the node C no longer refers to the classification label (as is the case with the Naive Bayesian classifier), but rather represents the *unobserved* cluster.

In order to fit a mixture model to a given data set, AutoClass employs the Expectation Maximization (EM) algorithm [41]. In this context, the cluster C that generated the data is treated as a *hidden*, or unobserved, variable with K values. EM iteratively estimates the distribution over the K values of the unobserved cluster variable C for each datum (thereby "completing" each datum) and then using this completed data D' to re-estimate the model parameters θ. The EM algorithm can be proven to converge to a parameter configuration $\tilde{\theta}$ which is a local maximum[1] with respect to the likelihood of the completed data. It is important to note that, as with other non-convex optimization processes, there generally exist many local maxima and the one that EM converges to is dependent on the initial, often randomly chosen, parameter configuration.

Again, the question arises of how K is chosen by AutoClass. While in many of our experiments we provide a fixed value of K to AutoClass, it is important to note that the algorithm does provide a means for selecting K automatically. This is accomplished by searching through the space of possible models (i.e., number of values for K), fitting one or several models for each choice of K, and then selecting the one which maximizes the following criterion

$$\begin{aligned} \log\ [\mathrm{P}(D' \mid M)\frac{\mathrm{P}(D \mid \tilde{\theta}, M)}{\mathrm{P}(D' \mid \tilde{\theta}, M)}] = \\ \log \mathrm{P}(D' \mid M) + \log \mathrm{P}(D \mid \tilde{\theta}, M) - \log \mathrm{P}(D' \mid \tilde{\theta}, M). \end{aligned} \tag{3.11}$$

It may initially seem that models with larger values for K will always be preferred since they provide more tunable parameters to better fit the data. However, the use of priors helps counterbalance this phenomena, as the introduction of more parameters causes the effect of the priors on the fit of the model to also become greater. At some point, the use of additional parameters will cause the increased influence of the priors

[1]To be more precise, the EM algorithm is only guaranteed to converge to a fixed point (i.e., where the gradient is zero), which can be either a local minimum, a local maximum, or a saddle point. Since the EM optimization procedure is performing a maximization, we simply refer to the convergence point as a "local maximum" in our expository text, although we note here that this statement is not precisely correct.

to prevent a better fit to the data from being achieved.

Once the model has converged, each document is now probabilistically assigned to each cluster (i.e., soft assignment) based on its final distribution over the variable C. Preliminary experiments (discussed further in Chapter 6) show, however, that in high-dimensional spaces such as text domains, the data usually have extreme probabilities (very near 1 and 0) for their cluster assignments, thus effectively emulating a hard assignment algorithm. Consequently, in our subsequent discussions of using AutoClass for text clustering, we actually assign documents to the cluster for which they have maximal probability as their degree of inclusion in any other cluster is generally negligible. A more complete discussion of variations of EM using both soft and hard assignment of data to clusters is discussed in [87].

Having completed a review of the probabilistic framework we use, as well as several machine learning algorithms which appear in subsequent chapters, we are now ready to conclude with the preliminaries. It is important to note, however, that there is a much longer history of research in machine learning than is possible to review here. Two of the earliest and most influential works that helped define the field are the books by Nilsson [121] and Duda and Hart [46], which both still serve as exceptional references. A collection of subsequent seminal papers in this field is found in [42]. Quinlan [130] presents a very readable survey of issues pertaining to supervised learning within the context of decision tree induction. Langley's [102] recent textbook on machine learning also presents an expository overview of the field. A more recent textbook by Mitchell [117] provides an excellent introduction to a number of different learning algorithms as well as a brief discussion of text classification. Finally, the European StatLog project [161] has compiled the results of extensive experimental comparisons of classification methods from the machine learning, statistics and neural networks communities into one volume. In addition to the empirical comparisons, this work highlights many of the salient issues pertaining to the different classification algorithms that are compared.

Chapter 4

Related Work in Information Access

To better understand the novel technical contributions encompassed in the development of SONIA, it is important to get an overview of related work in the area of Information Access. Given the long history of research in Information Access and the recent emergence of synergies between that community and researchers in Machine Learning, it is not possible for such an overview to be comprehensive. Rather, in this chapter, we point out the research most closely related with our own, and highlight seminal work in this area which has influenced our thinking.

4.1 Probabilistic Retrieval

Although we do not address the problem of ad hoc retrieval directly in this work, several research efforts aimed at this task are important to point out. Foremost, the seminal work of Salton, Wong and Yang [147] lays the ground work for the *Vector Space* model of documents and continues to be used today in modern retrieval systems. We also make use of this model (and examine it more fully in Chapter 2), as it aligns well with the representations used in the probabilistic machine learning methods we employ.

The use of probability theory for information retrieval problems has a long tradition, dating at least as far back as Maron's proposal for a probabilistic interpretation of "mechanized documentation" in the mid-1960's [113]. The role of probability was later defined more explicitly in the formulation of the *Probability Ranking Principle* by Robertson (who attributed it to W. S. Cooper) [134]. The principle states that, in order to optimize the effectiveness of an information retrieval system, documents matching a query should be ranked according to their probability of relevance to the user (i.e., the query). This principle thus suggests that methods for the estimation of probabilistic relevance of a document to a query are critical for the good performance of retrieval systems. Expanding on this notion, Cooper and Maron [33] proposed the idea of probabilistic indexing of documents in which index terms are given probabilistic weights based on the relevance of these index terms to queries likely to be given to the retrieval system. Much more recent work in probabilistic indexing by Fuhr [56, 57] also treats the retrieval problem as one of making probabilistic inferences about the relevance of documents to a query, and examines this problem from the different viewpoints of the query and the document.

The work along these lines closest in spirit to our own is that of van Rijsbergen [166], who essential boils the retrieval problem down to one of estimating the probability of observing each term in a document given that the document is relevant to the user's query. His formulation of this problem is exactly equivalent to a Bayesian approach to classification and thus bears strong similarities to the classification models we present in Chapter 8. In essence, his approach to document retrieval is simply to classify documents into either the category *relevant* or *non-relevant* using a Bayesian classifier. However, the applicability of such models to the retrieval task can be problematic as *a priori* it is unknown which documents are relevant to a query, making parameter estimation in these models difficult. Van Rijsbergen even extends his retrieval model to consider information about the probabilistic co-occurrence of words [165]. Nevertheless, his models still have more limited expressibility than those that we present later.

Other work in applying probabilistic models to the retrieval task includes the work of Turtle and Croft [163, 164] on using inference networks for document retrieval.

While this work claims to be casting the estimation of document relevance as a problem of inference in a Bayesian network [125], it is unclear the degree to which the system truly adheres to the principles of Bayesian inference. Still, this work has led to the development of the INQUERY system [19], one of the most successful retrieval tools at the TREC competition.

Fung and DelFavero [59] have also recently developed a retrieval system which does in fact operate in accord with Bayesian inference. This system, however, requires a good deal of manual knowledge engineering for constructing the network structure and selecting the appropriate index terms to represent within the network. Consequently, it points out yet another venue in which automated learning methods may be useful for information access, but we do not pursue that possibility here.

Further arguments in favor of the use of probability theory for retrieval problems are given by Cooper [32], who also provides a nice historical perspective on the subject. For more detailed information, we also refer the interested reader to the recently published collection of seminal works in information retrieval [155] that includes several papers on probabilistic models for retrieval.

Our account of previous work in using probabilistic retrieval models helps to show that such a formalism has not only been embraced by the IR community for its theoretical elegance, but that it has shown great empirical success for the retrieval task. As a result, we argue this formalism can also be of great benefit to addressing other problems in information access, to which we currently turn our attention.

4.2 Feature Selection for Text

As mentioned in Chapter 2, the application of feature selection is an important step in many information access systems. While stop word elimination and Zipf's-Law-based feature selection have been used for some time in retrieval systems, it is only in the last few years that more advanced forms of feature selection have been employed for other problems in information access.

One such approach to dimensionality reduction that has received quite a bit of attention in the information retrieval community recently is known as *Latent Semantic*

Indexing [40]. The idea here is to use singular value decomposition [159] to find an approximation to the original document vectors, such that these approximated document vectors lie in a lower dimensional subspace of the original vector space. The basis vectors of this low dimensional subspace (which are each linear combinations of the dimensions in the original vector space) then define the axes of the final vector space in which documents will be represented.

While LSI was originally proposed for retrieval, its use in clustering [149] and classification [47, 148] have been recently explored as well. In classification tasks, such a representation has shown some utility when used in conjunction with linear classifiers. This seem to follow from the fact that LSI, by creating feature vectors which are linear combinations of the original term space, is helping to capture some term co-occurrence information. Unfortunately, LSI does not seem to provide a compelling advantage for the clustering task.

We do not make use of LSI in our work, but rather choose to model the probabilistic co-occurrence of terms directly in our classification models. Moreover, one could argue that LSI, by convolving the problems of dimensionality reduction and term dependency modeling, creates a representation that is much harder to interpret. Instead, by separating the tasks of feature selection and term dependency modeling, we hope to not only create models which are more readily understandable, but also have a more modular architecture in which individual advances in feature selection or probabilistic modeling can be independently incorporated into our system.

More recent work in using feature selection specifically for document classification is seen in the work of Yang and Pedersen [175]. They compare a number of feature selection methods including simple document frequency schemes, as well as probabilistic approaches, such as computing the mutual information between each word and the topics in the corpus. As to be expected, they find that probabilistic approaches based on mutual information and chi-squared measures appear to give the best performance among the methods tested.

As we show in Chapter 7, our work on feature selection for classification builds on similar probabilistic methods for selecting features that have high information content relative to the classification task at hand. In fact, we show that the mutual information measure for feature selection is exactly equivalent to one particular variant of the feature selection method we have developed.

4.3 Document Clustering

There has been a good deal of previous work in the use of clustering technologies to help enable better information access. Early work along these lines was based on the *Cluster Hypothesis* which states that "closely associated documents tend to be relevant to the same requests" [166]. As a result, much of this research centered on the use of clustering as a means for improving retrieval speed and efficacy [167, 169]. The idea here is that if a collection of documents is clustered a priori, then queries need not be matched against individual documents, but only against some cluster representatives (e.g., the centroid or medoid of each cluster). The documents in the clusters matching most closely to the query would then be returned to the user. Not only could this speed-up retrieval, but it could also help retrieve relevant documents that did not contain the precise terms used in the user's query, but were part of a cluster of other documents that did.

Subsequently, a good deal of research in this area has focused on different methods for forming document clusters [131]. Willett [172] gives an excellent survey of much of this work up to the present decade. The past few years have seen new directions in document clustering focusing on more diverse issues such as models that allow documents to be full members of multiple clusters [141], as well as experimental studies examining the efficacy of using different document representations in conjunction with clustering [149].

Also in recent years, work in document clustering has turned toward using clustering to better enable user browsing of document collections. The work in this area most closely related to SONIA is the Scatter/Gather system [39] developed at Xerox Palo Alto Research Center. Studies with this system have shown that document clustering has the potential to help users quickly home in on the documents relevant to them and often successfully conveys a notion of the structure of a document collection to users [127]. Moreover, such a document clustering tool has also been used for navigating query results [70], and specialized user interfaces for querying and displaying document clusters have been developed for this system as a result [68]. The application of Scatter/Gather to query results has also lent further support for the Cluster Hypothesis, as it has been empirically observed that clustering tends to concentrate documents particularly relevant to a query in just one or two groupings [71]. Moreover, this work has shown that users are generally successful at locating a higher proportion of relevant documents by simply identifying the appropriate high-level groupings.

Other researchers have focused more on the use of visualization methods for conveying similarity between clustered documents to the user. For example, specialized interfaces have developed that allow users to navigate through the *dendogram* of documents generated by a hierarchical agglomerative clustering algorithm [3]. In this way, users can potentially locate subsets of particularly relevant documents. In much the same vein, other systems convey document similarity to the user via spatial layout [2]. Here, relevant documents are often located near each other (i.e., clustered) spatially. Thus, when a user locates a relevant document, it is more likely that they will find other relevant documents by examining documents in the local neighborhood. Although, this system does not explicitly form distinct clusters, the spatial layout of documents essentially forms de facto clusters.

Another example of the use of clustering to aid in information access is the use of *Self-Organizing Maps* (SOMs) [93] to group together related words into *word category maps*. Such maps are, in turn, used to automatically organize (i.e., cluster) documents according to the words that they contain. Early work along these lines was conducted by Lin, Soergel and Marchionini [112]. Subsequently, Timo *et al* developed the WEBSOM system [162] which provides an interface that allows users to navigate a word category map and zoom in on groups of documents related to a particular group of words. The SOM approach seems to require a fairly sizable initial corpus to generate a useful word category map, however, and thus may not be directly applicable to query results.

Recently, the SenseMaker interface [11] has been developed, which allows for documents returned in response to a query to be "bundled" (their term for "clustered") according to various criteria. Since this system is designed for use primarily with Web documents, the bundling criteria include simple non-content document features, such as identical URL's or identical URL domains. To extend the functionality of this interface to include full-text clustering of documents, we have connected some of the clustering and classification technology developed as part of SONIA with SenseMaker [142]. Still, since SenseMaker does not allow for the explicit formation and display of cluster hierarchies, we have developed a separate user interface for SONIA which is described in Chapter 10.

As seen in much of the more recent work above, the use of document clustering is often closely tied with the development of various interfaces for information access. It is important to note that although we have provided our own novel interface as part of SONIA, this is not one of our primary research foci. Rather, by concentrating on improved methods for clustering, we believe that advances in the underlying clustering technology can be of benefit to any interface for information access that makes use of these improvements. Still, we believe the issue of interface design is very important in achieving better information access, and while we do not address this issue further here, we refer the interested reader to Hearst's recent overview of this area [67].

4.4 Document Classification

Spurred on by the growing availability of on-line information, research in document classification has grown enormously in the past few years. One of the earliest and still used document classification methods, now called the Rocchio algorithm [137] after its inventor, is based directly on the Vector Space model for retrieval. This algorithm essentially creates a binary classifier by summing together the term vectors of one class of documents, while subtracting from this sum the term vectors for the other class of documents. This procedure produces a final weight vector which can then be used to classify new document vectors by computing a dot product and thresholding appropriately.

More recently, several traditional machine learning techniques have been applied to the problem of document classification. For example, decision tree induction [130] has been explored as a means of text classification [37]. Likewise, Wiener, Pedersen and Weigand [171] have employed neural networks [138] for topic spotting, which can be seen as a form of classification problem. This research has been important in pointing out the strengths and limitations in applying these methods to text. Other methods, however, have come into favor of late.

Masand, Linoff and Waltz [114], for example, have used *Memory Based Reasoning* [157] (essentially a form of the nearest neighbor algorithm [46]) to effectively classify news stories given a very large number of training documents on a highly parallelized machine. Further evidence of the efficacy of the nearest neighbor algorithm is found in the work of Yang and Chute [174], who use this method in conjunction with different word weighting schemes for text classification.

In a different vein, rule induction algorithms have recently become one of the more popular methods for text classification. Such algorithms induce a set of logical rules that predict a classification label for a document given that some set of words appears in the document. The rule induction system of Apte, Damerau and Weiss [7] has been successfully applied to the classification of Reuters newswire stories. Likewise, Cohen has developed the RIPPER algorithm [28] which contains extensions, such as using set-valued features, that make it particularly applicable to text domains. Riloff and Lehnert [133] have also examined learning systems for text classification, focusing on using hand-coded domain knowledge to help induce rules whose antecedents are based on more structured information extracted from text than simple word occurrences.

Of closest relation to our own work is previous research in probabilistic text classification methods. Lewis, in his seminal thesis on text classification [106], showed that the Naive Bayesian classifier [64] (which we discuss in detail in subsequent chapters) is often quite effective for use in text domains, even though the strong statistical independence assumptions it makes are often clearly violated in practice. Further support for this point is also seen in Lewis and Ringuette's [110] empirical work comparing such simple probabilistic classifiers with decision trees that are able to capture the probabilistic dependencies between words in a document. Lewis and Gale [109] also explore methods for reducing the amount of data necessary to train a Naive Bayesian classifier. More recently, the Web Knowledge Base project at Carnegie Mellon University [36] has also made extensive use of the Naive Bayesian classifier as a means of categorizing Web pages in order to then apply category-specific information extraction tools.

It is important to note that the term "Naive Bayesian classifier" has been used in the text classification community to refer to two different classifier models. While these classifiers are both similar in their use of Bayesian decision making and strict independence assumptions, they differ in their underlying event models. As explained by Lewis [108] and McCallum and Nigam [115], one such model represents the probability of word appearances using multiple independent Bernoulli trials (as explained in Chapter 3 and seen in our work [95]). Alternatively, word appearances can also be represented using multiple draws from a single multinomial distribution over words (as described in Mitchell's *Machine Learning* text [117]). In subsequent chapters of this work, we describe and make use of the multiple Bernoulli model as it lends itself to extensions such as modeling word dependencies [139] which are problematic to incorporate in the single multinomial model. Still, it is important to realize that while these underlying differences do exist, good performance has been witnessed using both event models.

Given that so many methods for text categorization have been developed and explored, a number of extensive studies comparing different algorithms and document representations have also emerged. Schuetze, Hull and Pedersen [148] provide an excellent comparison of classification methods including logistic regression, linear discriminant analysis and neural networks. They also compare using a simple term weighting representation documents with Latent Semantic Indexing [40] and find that a redundant combination of these representations seems to provide the best performance in their experiments.

On a related note, Lewis *et al* [111] compare a number of learning methods for linear text classifiers including Rocchio's algorithm, the Widrow-Hoff (WH) update rule [170] and the exponentiated gradient (EG) algorithm [89]. They find that both WH and EG yield consistently superior results to Rocchio. Furthermore, they point out that since EG seems to drive many of the linear discriminant coefficients to zero, effectively reducing the number of features used in making classification decisions, it may be possible to augment the feature space of such models with more expressive features (i.e., multi-word combinations) to produce better results; however, they do not actually conduct such experiments.

Addressing this issue, Cohen and Singer [29], as respective advocates of rule-based and probabilistic learning methods, provide a nice comparison of the RIPPER rule inducer with the probabilistic Sleeping Experts algorithm. Their results provide a clear indication that using what they call "context" (i.e., multi-word predictive features) in text classification can yield significant improvements in accuracy. Such results lend further credence to the fact that there is room for improvement over linear classification methods, including Naive Bayes, that do not consider word co-occurrence information. We explore this issue more fully in Chapter 8.

Exploring an entirely different learning framework, Joachims [82] provides a comparison of the recently popularized Support Vector Machine (SVM) [168] with several previously used learning methods for text classification, including Naive Bayes, decision trees and nearest neighbor methods. He concludes that SVMs produce the best overall results in terms of precision and recall measures. Unfortunately, Joachims optimizes some of the parameters of his learning methods after examining their performance on the testing data, which raises some questions about the validity of the absolute performance numbers he reports. Still, his relative ranking of methods is likely to be trustworthy.

Dumais *et al* [48] also compare SVMs with a Rocchio-like classification method (called *Find Similar*), Bayesian classifiers, and decision trees. They also report that Support Vector Machines provide the best overall results (without using the testing data in any way during the learning algorithm parameter optimization process). Still, recognizing the importance of performance evaluation, the desire to also have understandable probabilistic semantics for such models has led researchers toward initial attempts to understand SVMs in a probabilistic framework [62, 38, 73]. This is an area of very recent inquiry and not one that we delve further into here. Rather, staying well within the realm of probabilistic learning, we explore methods whose underlying assumptions align better with what we believe to be true about text domains (i.e., probabilistic dependencies between words exist). We believe that a fruitful venue for future work will be to establish stronger connections between our models and the Structural Risk Minimization framework on which SVMs are based, but that is beyond the scope of the work presented here.

In addition to the wide variety of comparative studies of document classification, there are also numerous successful research applications of such technology that are worth pointing out. We present these applications to show that our work on improved methods for document classification hold promise not only in the context of our system SONIA, but also in a variety of other stand-alone applications.

Lang's NewsWeeder system [101] shows one application of classification to Usenet Newsgroup article routing. After learning from user judgments of previously read news articles, the system scours through a wide variety of newsgroups collecting other relevant articles for the user. Similarly, Balabanovic has developed a system, known initially as LIRA and later as Fab [9, 8], that works in much the same fashion for recommending potentially interesting Web pages to users. Fab's underlying technology differs from NewsWeeder in that it makes use of a collection of distributed agents to learn multiple topics of interest for each user of the system. Also working in the context of the Web, Pazzani and Billsus have developed a system very similar in spirit to Fab, named Syskill and Webert [123, 122]. This system creates a "user profile" by training a Naive Bayesian classifier on a user's rating of various Web pages and then uses this classifier to identify previously unseen Web sites that may be of interest to the user. In much the same vein, the WebWatcher system of Joachims, Freitag and Mitchell [83] also uses document classification technology to learn topics of interest to a user. The system then acts as a tour guide for the Web, recommending links for a user to follow *in situ* on the Web pages which they are browsing.

Lastly, considering the problem of electronic mail classification, Cohen [27] has used rule-based learning methods for automatically classifying messages into appropriate folders in a user's E-mail archive. Tackling a similar problem, we have, in collaboration with several colleagues, also applied the probabilistic classification methods described in later chapters to successfully filter out junk E-mail (known colloquially as "spam") from a user's incoming mail stream [140]. While we do not give a full account of this still on-going work here, one point of particular interest that stems from this work is that additional features representing non-textual domain-specific information (such as a message sender's E-mail domain type) can be easily integrated into our feature selection and classification models (described in Chapters 7 and 8, respectively) without requiring any modification to the underlying algorithms. The use of such additional information led to a dramatic improvement in the empirical performance of these methods in the junk mail filtering task and points out the potential benefits of considering a combination of textual and non-textual features for domain-specific applications in future work.

Part II

Clustering

Chapter 5

Feature Selection for Clustering

5.1 Introduction

Beginning with the task of document clustering, we seek to group together documents that possess strong internal consistencies. As outlined in Chapter 1, such an *unsupervised* learning task is useful for a wide variety of problems, including helping users more easily browse the topics contained in a large document collection [39, 143]. Moreover, clustering has been successfully applied to a broad range of data mining problems in a number of different domains [23], helping to uncover important substructure within the data.

Let us begin by considering the AutoClass algorithm presented in Chapter 3. While such a mixture modeling method may be quite effective for clustering low-dimensional data, it may be problematic to apply directly in very high-dimensional spaces. For example, text collections can have on the order of 10^3 to 10^4 features (even after applying the Zipf's-Law-based dimensionality reduction technique described earlier), making them quite challenging for mixture modeling.

To address the issue of high dimensionality, we present methods for unsupervised feature selection that seek to maintain as much information from the original probability distribution over the data as possible. The intuition here stems from the idea that mixture modeling often seeks to capture the probabilistic dependencies that exist between features in a data set. This is accomplished by assuming, as in AutoClass,

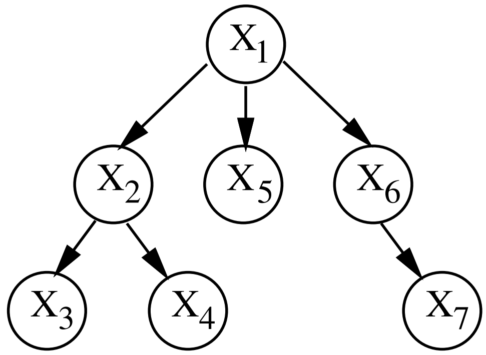

Figure 5.1: A tree-structured dependency model.

that all features in the model are independent when conditioned on the mixture component C. Consequently, any dependencies between features must be captured via the states of the mixture variable. As a result, if we eliminate the features that have the least effect on the distribution of the remaining features, we can focus the mixture model on the strongest probabilistic dependencies in the data without getting bogged down by the estimation of many more parameters from the data. The hope, then, is that the reduction in the parameter space will help control the variance associated with estimating many parameters and will thus produce better models in the end. Note that we will use the terms *Bayesian network structure* and *dependency model* interchangeably.

In the remainder of this chapter, we give a formal presentation of our probabilistic framework and follow with several theoretical results pertaining to minimizing information loss during feature selection. Based on these observations, we then present two feature selection algorithms and give empirical results using them on high-dimensional text data.

5.2 Mixture Modeling Revisited

Recall that in a mixture model, a distribution can be encoded as a Bayesian network which contains a *hidden* variable C corresponding to "clusters" that data instances may belong to. This cluster variable then influences the distributions of all other features in the network. Since, in AutoClass, an underlying *Naive Bayes* dependency model (depicted in Figure 3.2) is used, it becomes clear from the structure of this model that any feature interactions in the data must be captured via the states of the cluster variable C. Thus, the resulting clusters will generally be most heavily influenced by the strongest feature dependencies that exist in the data being modeled.

We believe that generating clusters based on such feature dependencies makes intuitive sense for document collections, since we would expect that documents about a given topic would often contain several strongly inter-dependent words. Unfortunately, in cases where there are very many features in the mixture model, as is often the case with document clustering, the probabilistic dependencies that exist between features can become more problematic to discover. This problem results from the fact that the space of possible models is enormous, and some of the dependencies that exist in the data may be swamped by noise from a large number of uninformative features.

Still, we can make use of the intuition that the clusters discovered via mixture modeling rely on capturing the strongest word dependencies in a collection. Hence, we believe that by identifying the words that are *least* correlated with other words, we are likely to find good candidate features to eliminate from subsequent mixture models. Armed with this insight, we now turn our attention to considering the general problem of modeling a data set using a Bayesian network *without* a hidden cluster variable (i.e., density estimation), so as to more directly model probabilistic dependencies between features in the data.

Recalling that the independence assumptions encoded in a Bayesian network allow us to factor a probability distribution into components that are computationally more manageable and more robust for parameter estimation, we seek to limit the number of dependencies allowed in such models. Still, if these independence assumptions are too severe, we may sacrifice too much model expressivity for tractability.

In previous work on learning such networks from data, it has been observed that *tree-structured* Bayesian networks provide a good compromise between expressivity and computational tractability. Indeed, the first explorations of such tree-structured dependency models dates back over 30 years under the rubric of *optimal dependence* *trees* [26]. More recently, Friedman, Geiger, and Goldszmidt [52] show very good performance with such tree-structured dependency models for supervised classification tasks, and their use in other contexts has been examined by Geiger [60].

More formally, a tree-structured Bayesian network decomposes a distribution $\mathrm{P}(\boldsymbol{X})$ into the product $\prod_i \mathrm{P}(X_i \mid S_i)$, where S_i denotes the *single* parent node of X_i or is $\emptyset$ if X_i has no parent. An example of such a tree-structured dependency model is shown in Figure 5.1.

Using such tree-structured dependency models as a starting point, we can develop methods for feature selection which can minimize the information loss with regard to the distribution being modeled. By reducing the dimensionality of the data, we hope to improve the empirical performance of our mixture models by reducing the variance associated with the estimation of a large number of parameters. With this goal in mind, we present a series of theoretical results which pave the way for well-founded and practical algorithms for unsupervised feature selection.

5.3 Theoretical Underpinnings

We first begin by defining terms and our loss criterion. Let $\mathrm{P}(\boldsymbol{X})$ define an arbitrary probability distribution over the features $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ and let $\mathrm{P}_T(\boldsymbol{X})$ denote a probability distribution over $\boldsymbol{X}$ which is defined by a tree-structured dependency model. Moreover, we employ the information-theoretic measure of relative entropy (also known as Kullback-Liebler divergence [100]) to measure the loss in approximating one distribution with another. More formally, let μ and σ be two distributions over some probability space Ω. The *relative entropy* of μ to σ is defined as

$$D(\mu, \sigma) = \sum_{x \in \Omega} \mu(x) \log \frac{\mu(x)}{\sigma(x)} \ . \tag{5.1}$$

Note that the roles of μ and σ are not symmetrical in this definition. Generally speaking, the idea is that μ is the "right" distribution, and σ is our approximation to it. Then, $D(\mu, \sigma)$ measures the extent of the "error" that we make by using σ as a substitute for μ.

We now show that an optimal tree-structured dependency model for a distribution (with respect to the relative entropy loss criterion defined above) can be constructed efficiently. Chow and Liu [26] in their seminal work on optimal dependence trees, present a simple and efficient procedure for determining the structure of an optimal dependence tree. We present an overview of their algorithm here, and follow it with their main theoretical result.

The Chow and Liu procedure for finding an optimal dependence tree can be summarized as follows. Let G be a complete undirected graph, where each node in G represents a single feature X_i in $\boldsymbol{X}$. Let each arc between two feature nodes X_i and X_j in G be weighted by the mutual information $MI(X_i; X_j)$ between those features. Let MST be the maximal spanning tree of G. Direct all the arcs in MST "away" from some arbitrarily chosen node, thereby producing an optimal dependence tree.

Theorem 1 (Chow and Liu, 1968 [26]) *The distribution $P_T(\boldsymbol{X})$ defined by the tree-structured dependency model MST minimizes $D(P(\boldsymbol{X}), P_T(\boldsymbol{X}))$ over all distributions defined by tree-structured dependency models.*

We omit a review of the proof of this theorem for the sake of brevity. Still, we note that constructing such a maximal spanning tree can be done in time $O(n^2 m)$ [34], where n is the number of features and m is the number of data instances.

The directed maximal spanning tree MST obviously defines the structure of a tree-structured Bayesian network. Hence, we associate each node of the graph MST with a conditional probability table representing the probability for the feature represented by that node, conditioned on the (maximum of one) parent feature of the node. We denote the distribution defined by this dependency model as $\mathrm{P}_{MST}(\boldsymbol{X})$.

Now, let $\mathrm{P}_F(\boldsymbol{X})$ be any distribution whose underlying dependency graph (i.e., Bayesian network), denoted F, is a directed forest. Let $\mathrm{P}_{F_{MST}}(\boldsymbol{X})$ be a distribution whose underlying Bayesian network F_{MST} is a directed forest whose edges are some subset of the edges in MST. Moreover, the conditional probability tables associated with the nodes in F_{MST} are identical to the ones associated with MST, except for the cases where nodes that had a parent in MST are parentless in F_{MST}. These parentless nodes in F_{MST} are associated with a marginal probability table with is determined by marginalizing the conditional probability table in corresponding node in MST. Finally, let $\boldsymbol{X}_{-i}$ denote the set of features $\boldsymbol{X} - \{X_i\}$.

Theorem 2 *Deleting the arc u with the least mutual information weight in a dependency forest F yields a dependency forest F' such that $D(P_F(\boldsymbol{X}), P_{F'}(\boldsymbol{X}))$ is minimized over all choices of u.*

Proof:

$$D(\mathrm{P}_F(\boldsymbol{X}), \mathrm{P}_{F'}(\boldsymbol{X})) = \sum_{\boldsymbol{x}} \mathrm{P}_F(\boldsymbol{x}) \log \frac{\mathrm{P}_F(\boldsymbol{x})}{\mathrm{P}_{F'}(\boldsymbol{x})} \tag{5.2}$$

$$= \sum_{\boldsymbol{x}} \mathrm{P}_F(\boldsymbol{x}) \log \prod_i \frac{\mathrm{P}_F(x_i \mid S_i = s_i)}{\mathrm{P}_{F'}(x_i \mid S'_i = s'_i)} \tag{5.3}$$

where S_i is the parent of X_i in F and S'_i is the parent of X_i in F'. Recall that S_i (S'_i, respectively) will contain at most one feature since F (F') is a forest, and may be $\emptyset$ if X_i has no parent. Consider an arc u that connects X_k with S_k, and let F' be the forest obtained from F by dropping u. Consequently, $S'_k = \emptyset$ and $S'_i = S_i\ \forall i \neq k$. Incorporating this into Eq. 5.3 yields:

$$\sum_{\boldsymbol{x}} \mathrm{P}_F(\boldsymbol{x}) \log \frac{\mathrm{P}_F(x_k \mid s_k)}{\mathrm{P}_{F'}(x_k)} = \sum_{\boldsymbol{x}} \mathrm{P}_F(\boldsymbol{x}) \log \frac{\mathrm{P}_F(x_k \mid s_k)}{\mathrm{P}_F(x_k)} \tag{5.4}$$

$$= \sum_{x_k, s_k} \mathrm{P}_F(x_k, s_k) \log \frac{\mathrm{P}_F(x_k \mid s_k)}{\mathrm{P}_F(x_k)} \tag{5.5}$$

$$= MI(X_k; S_k) \tag{5.6}$$

Since the arc u between X_k and S_k is weighted by $MI(X_k; S_k)$, dropping the arc for which this value is smallest thus minimizes $D(\mathrm{P}_F(\boldsymbol{X}), \mathrm{P}_{F'}(\boldsymbol{X}))$. ∎

Corollary 3 *Deleting the least weighted arc u in a directed maximal spanning tree MST yields a directed forest F_{MST} such that $D(P_{MST}(\boldsymbol{X}), P_{F_{MST}}(\boldsymbol{X}))$ is minimized over all choices of u.*

Proof: Since a tree is a forest by definition, this result follows immediately from Theorem 2 and the definition of F_{MST}. ∎

While the theorem above gives us insight as to how to iteratively drop arcs from a Bayesian network while minimizing information loss in the distribution being modeled, we are also interested in dropping nodes from a Bayesian network so as to reduce the total number of features in a model. To that end, we consider which features (i.e., nodes) we may delete from a Bayesian network which will have the least impact on the distributions of the remaining features.

Theorem 4 *Deleting a node representing feature X_i in an arbitrary directed dependency graph G causes no loss with respect to the marginalized relative entropy given by $\sum_{x_i} P_G(x_i) \cdot D(P_G(\boldsymbol{x}_{-i} \mid x_i), P_G(\boldsymbol{x}_{-i}))$ if X_i is unconnected to any other node in G.*

Proof:

$$\begin{aligned} & \sum_{x_i} \mathrm{P}_G(x_i) \cdot D(\mathrm{P}_G(\boldsymbol{x}_{-i} \mid x_i), \mathrm{P}_G(\boldsymbol{x}_{-i})) \\ = & \sum_{x_i} \mathrm{P}_G(x_i) \cdot \sum_{\boldsymbol{x}_{-i}} \mathrm{P}_G(\boldsymbol{x}_{-i} \mid x_i) \log \frac{\mathrm{P}_G(\boldsymbol{x}_{-i} \mid x_i)}{\mathrm{P}_G(\boldsymbol{x}_{-i})} \end{aligned} \tag{5.7}$$

Since X_i is unconnected to any other node in the dependency graph G, $\mathrm{P}_G(\boldsymbol{x}_{-i} \mid x_i) = \mathrm{P}_G(\boldsymbol{x}_{-i})$. Substituting this result into Eq. 5.7 yields:

$$\sum_{\boldsymbol{x}} \mathrm{P}_G(\boldsymbol{x}) \log \frac{\mathrm{P}_G(\boldsymbol{x}_{-i})}{\mathrm{P}_G(\boldsymbol{x}_{-i})} = 0. \tag{5.8}$$

▮

In an attempt to generalize beyond tree-structured dependency models, we also provide a more general result, but here need to consider a different form of approximation on the underlying dependency model. We begin with an observation about the mathematical relationship of the marginalized relative entropy to the mutual information between pairs of features.

Observation 5 *The marginalized relative entropy $\sum_{x_i} P(x_i) \cdot D(P(\boldsymbol{x}_{-i} \mid x_i), P(\boldsymbol{x}_{-i}))$ is equal to the sum of conditional mutual information values $\sum_{j=1, j \neq i}^{n} MI(X_i; X_j \mid X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{j-1})$.*

Proof:

$$
\begin{aligned}
&\sum_{x_i} \mathrm{P}(x_i) \cdot D(\mathrm{P}(\boldsymbol{x}_{-i} \mid x_i), \mathrm{P}(\boldsymbol{x}_{-i})) \\
&= \sum_{x_i} \mathrm{P}(x_i) \cdot \sum_{\boldsymbol{x}_{-i}} \mathrm{P}(\boldsymbol{x}_{-i} \mid x_i) \log \frac{\mathrm{P}(\boldsymbol{x}_{-i} \mid x_i)}{\mathrm{P}(\boldsymbol{x}_{-i})} && (5.9) \\
&= \sum_{\boldsymbol{x}} \mathrm{P}(\boldsymbol{x}) \log \frac{\mathrm{P}(\boldsymbol{x}_{-i} \mid x_i)}{\mathrm{P}(\boldsymbol{x}_{-i})} && (5.10) \\
&= MI(X_i; \boldsymbol{X}_{-i}) && (5.11) \\
&= \sum_{j=1, j \neq i}^{n} MI(X_i; X_j \mid X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{j-1}) && (5.12)
\end{aligned}
$$

∎

Here, we note that the summation in Eq. 5.12 is intractable to compute when the number of features n is large, since the size of the conditional probability tables needed to compute each mutual information term grows exponentially with the number of conditioning variables. Alternatively, we can consider a simple approximation to this sum that is easy to compute. In this approximation, we ignore the conditioning variables in each mutual information term, giving us

$$
\sum_{j=1, j \neq i}^{n} MI(X_i; X_j \mid X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{j-1}) \approx \sum_{j=1, j \neq i}^{n} MI(X_i; X_j) \tag{5.13}
$$

Note that this approximation is actually only a true equality in cases where all variables X_i are independent of each other. Still, if we believe that many of the features in our domain do not exhibit significant "higher order" interactions and, hence, we may capture most of the variable dependencies in our domain via simple pair-wise interactions, then this approximation may be a reasonable one. Moreover, using this approximation allows us to relax our restriction of only considering tree-structured models when attempting to capture the probabilistic dependencies in a domain. The degree to which this approximation is valid is, thus, mainly an empirical question for a particular data set that is being modeled. We explore this issue further in Section 5.5.

5.4 Feature Selection Algorithms

The theoretical results presented above, give rise to two feature selection algorithms outlined below. These algorithms are based on the insight that features that appear to only weakly influence the remaining domain variables are good candidates for elimination, prior to building a mixture model.

It must be stressed, however, that using a tree-structured dependency model to measure the degree to which a feature X_i influences the rest of the features in the domain is only an approximation to the true degree of feature interaction in the domain. Consequently, eliminating the variable X_i that minimizes $\sum_{x_i} \mathrm{P}_T(x_i) \cdot D(\mathrm{P}_T(\boldsymbol{x}_{-i} \mid x_i), \mathrm{P}_T(\boldsymbol{x}_{-i}))$, where $\mathrm{P}_T(\cdot)$ is a probability distribution defined by a tree-structured dependency model, may not also minimize $\sum_{x_i} \mathrm{P}(x_i) \cdot D(\mathrm{P}(\boldsymbol{x}_{-i} \mid x_i), \mathrm{P}(\boldsymbol{x}_{-i}))$, where $\mathrm{P}(\cdot)$ is the true distribution for the domain. Still, if we believe that the tree-structured dependency model is a good approximation, then we may be confident that in many cases we are indeed eliminating the features that have the least impact on the rest of the domain features.

With that said, we now present a feature selection algorithm based on building a tree-structured dependency model of a given domain.

`Procedure MST-FeatureSelection`

1. $\forall i, j \;\; i \neq j$ `Compute` $MI(X_i, X_j)$
2. `Build maximal spanning tree` MST `of complete graph` G `where the arc between each pair of nodes` X_i `and` X_j `is weighted by` $MI(X_i, X_j)$.
3. $F_{MST} \leftarrow$ `direct-arcs(`MST`)`, $\boldsymbol{R} \leftarrow \boldsymbol{X}$.
4. `while (`$|\boldsymbol{R}| > k$`) do`
    - 4.1. `if` $\exists v$ `where` v `is a node representing feature` X_d `that is not connected to any other node in` F_{MST}
        - 4.1.1. `then` $\boldsymbol{R} \leftarrow \boldsymbol{R} - \{X_d\}$, $F_{MST} \leftarrow F_{MST} - v$.
        - 4.1.2. `else remove least weighted arc from` F_{MST}.

In the `MST-FeatureSelection` algorithm, we begin by constructing the optimal tree-structured dependency model (according to Thm. 1). Then, we iteratively delete the least weighted arcs in the dependency tree (forest) and eliminate features corresponding to nodes that become isolated as a result (in accordance with Thms. 2, 3 and 4). We continue to eliminate nodes in this fashion until only k features remain, where k is a user-set parameter.

Alternatively, we may consider a feature selection algorithm which does not make use of an underlying tree-structured dependency model. Here, we use the assumption (formalized in Eq. 5.13) that the influence a feature X_i exerts on the rest of the features in the domain can be measured as the sum of the unconditional pairwise mutual information values between X_i and every other feature in $\boldsymbol{X}$. Since such an approximation requires that each pair of features be independent of all remaining features, we say that our algorithm based on this assumption is *Pairwise Interaction Limited*, or *PIL* for short.

`Procedure PIL-FeatureSelection`

1. $\forall i, j \;\; i \neq j$ `Compute` $MI(X_i, X_j)$.
2. $\boldsymbol{R} \leftarrow \boldsymbol{X}$.
3. `while` $(|\boldsymbol{R}| > k)$ `do`
    3.1. $\forall X_i \in \boldsymbol{R}$ `Compute` $\alpha_i \leftarrow \sum_{X_j \in \boldsymbol{R} - \{X_i\}} MI(X_i; X_j)$
    3.2. $X_d \leftarrow$ `feature with minimal` α_d.
    3.3. $\boldsymbol{R} \leftarrow \boldsymbol{R} - \{X_d\}$.

The `PIL-FeatureSelection` algorithm (based directly on Obs. 5 and Eq. 5.13) is also iterative in nature, eliminating features which have minimal summed mutual information with all other remaining nodes in the dependency graph. Since we believe intuitively that a feature will often not depend on all other features in the model, however, we can also consider a heuristic variant of this algorithm where in Step 3.1 we compute the sum over just the q features which have maximal mutual information with each feature X_i. To test the efficacy of these algorithms, we presently give a number of empirical results using these algorithms on high-dimensional text data.

Data set	Number of Docs	Number of Words	Categories	Majority Accuracy
D1	486	1143	Natural Gas, Soybean, Dollar	46.7%
D2	466	1001	Gold, Coffee, Sugar	41.0%
D3	289	552	T-Bill, Yen, Reserves	43.9%
D4	467	1126	GNP, Livestock, Sugar	39.6%
D5	1426	1953	Loan, Interest, Money Effects	42.9%

Table 5.1: Data sets used in experiments.

5.5 Empirical Results

Our experiments using these feature selection algorithms are conducted on various subsets of the Reuters-22173 data set [132]. The data sets used here are created by selecting a subset of the class labels from the Reuters collection, and then including all of the documents that are assigned only one of the labels in this subset. As mentioned previously, we pre-processed this data in accordance with Zipf's Law to eliminate any words that appeared fewer than 10 or greater than 1000 times, as providing too little discriminating power between documents. Thus, we sought to create a more challenging problem by previously eliminating features that we would expect not to be useful based on the fact that they are extremely rare or common.

A description of these data sets is given in Table 5.1. This table includes the number of documents (instances) in each data set, the number of distinct words (features) *after* applying the Zipf's Law-based reduction, the names of the topic present in each collection, and the classification accuracy that would be realized if all the documents in the data set were classified into the majority class.

In order to evaluate our feature selection algorithms, we use previously labeled data and measure how well mixture modeling (i.e., clustering) recovers the known label structure in the data. The learning algorithm, however, is given no information about the true (i.e., human assigned) label of each document. We train a mixture model on each data set using AutoClass. After training is completed, we designate the predicted label for all documents in each cluster to be the true label that the majority of documents in that cluster have. Once we have an actual and predicted label for each document, we can now simply compute the classification accuracy as the percentage of documents whose predicted and actual labels match.

We note that this method for evaluating the accuracy of clustering produces a useful and intuitive score. This accuracy measure is also especially well-suited to the task of grouping documents in a semantically meaningful manner, which would be the aim of applying mixture modeling to text data. Moreover, we did not simply evaluate the marginal likelihood of the resulting mixture models since such a comparison is not meaningful over data with different numbers of features.

We recognize the difficulties of quantitatively evaluating clustering in general. And, we note that our method for evaluating the accuracy of a clustering may not be an ideal evaluation method in all situations, especially since it tends to favor larger numbers of clusters. Indeed, there currently appears to be no universally accepted method for evaluating clustering results. Still, we believe that in our controlled experimental setting, the advantages of this measure outweigh its shortcomings for the task that we are addressing.

As part of our experimental methodology, we first build a mixture model using the full feature set (after the Zipf's Law reduction) and we refer to this as the *Baseline*. We then use the `MST-FeatureSelection` algorithm, denoted *MST*, to eliminate features and then train a new mixture model. We likewise evaluate `PIL-FeatureSelection` using both the original (sum over all features) formulation, which we denote *PIL*, and the heuristic formulation (summing over the five maximal mutual information values for each feature), denoted *PIL-5*. Finally, we also consider applying *supervised* feature selection to the data, denoted *Supervised*, which involves eliminating features which have the smallest mutual information with the true *class* variable. We can consider this as a rough upper bound for the efficacy of feature selection, as this feature selection method has direct access to the class variable that the mixture model is trying to discover. So applying such a feature selection method should yield features which more directly guide the mixture model to discovering the different groupings in the data. Note that in this case, the mixture model itself, obviously, is still not given direct access to the class variable.

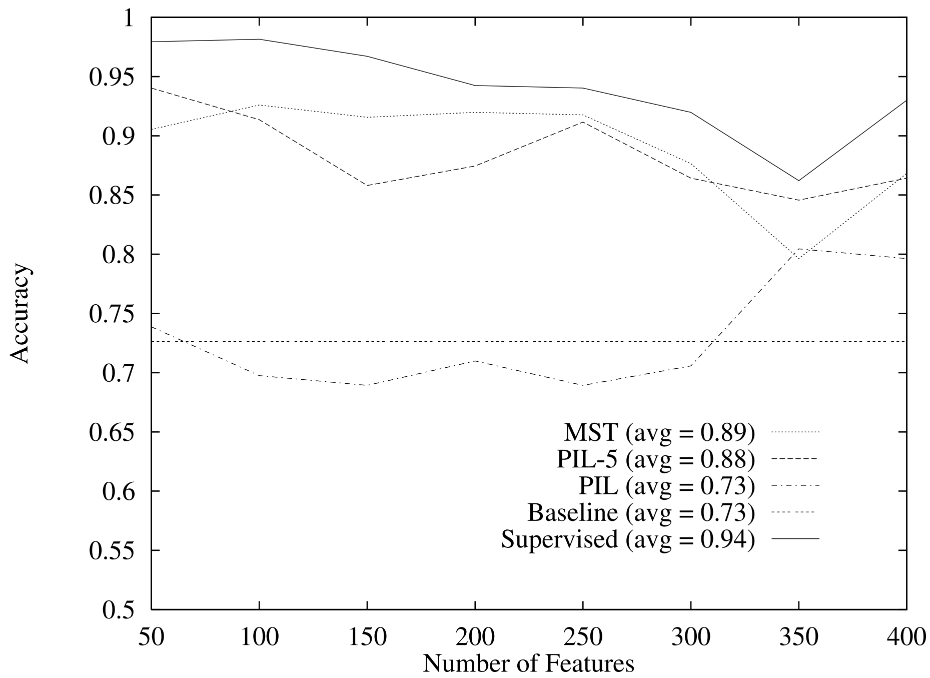

Figure 5.2: Results on Dataset 1.

The results of these experiments across feature sizes from 50 to 400 are reported in Figures 5.2 to 5.6. Note that the *Baseline* does not actually involve feature selection, but is simply presented as a straight line in the figures for visual comparison. The averages given in the legends are the average accuracies over the entire range from 50 to 400 features (in increments of 50). Since our evaluation measure favors greater numbers of clusters, we did a "post mortem" analysis to measure the degree to which the number of clusters varied between runs. In virtually all cases, the number of clusters produced on the original (non-reduced) data was equal to or greater than the number of clusters discovered in the data after feature selection, indicating that the improvement seen in using our feature selection algorithms is not a result of causing AutoClass to form more clusters on the reduced data.

From the results in Figures 5.2 to 5.6, it is evident that both the *MST* and *PIL-5* methods for feature selection produce consistently better results than mixture modeling applied to the original data. This is especially important when we realize that in several cases these methods are reducing the feature set to less than 5% of its

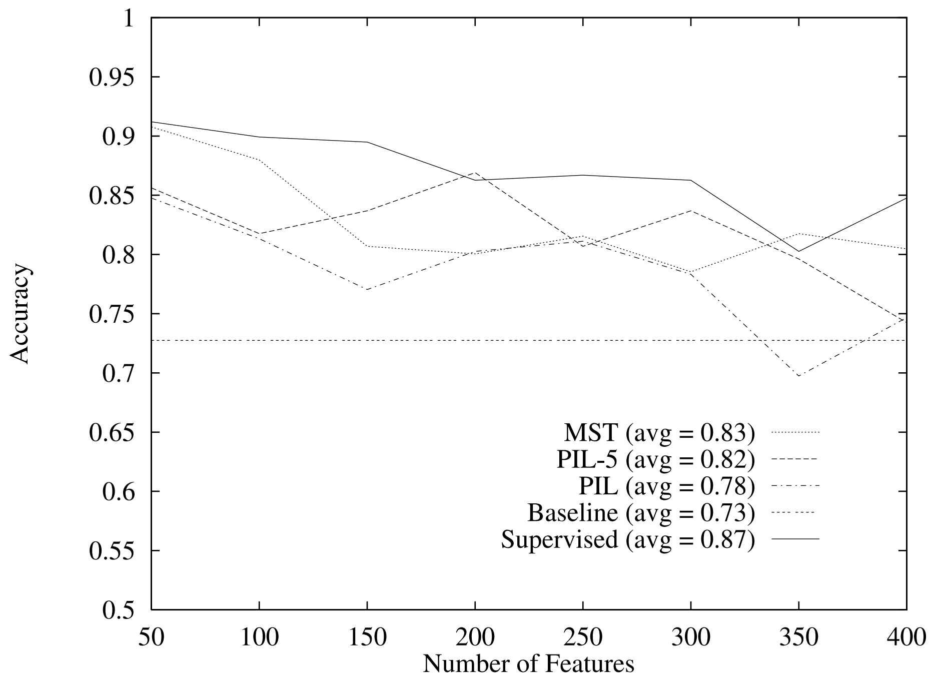

Figure 5.3: Results on Dataset 2.

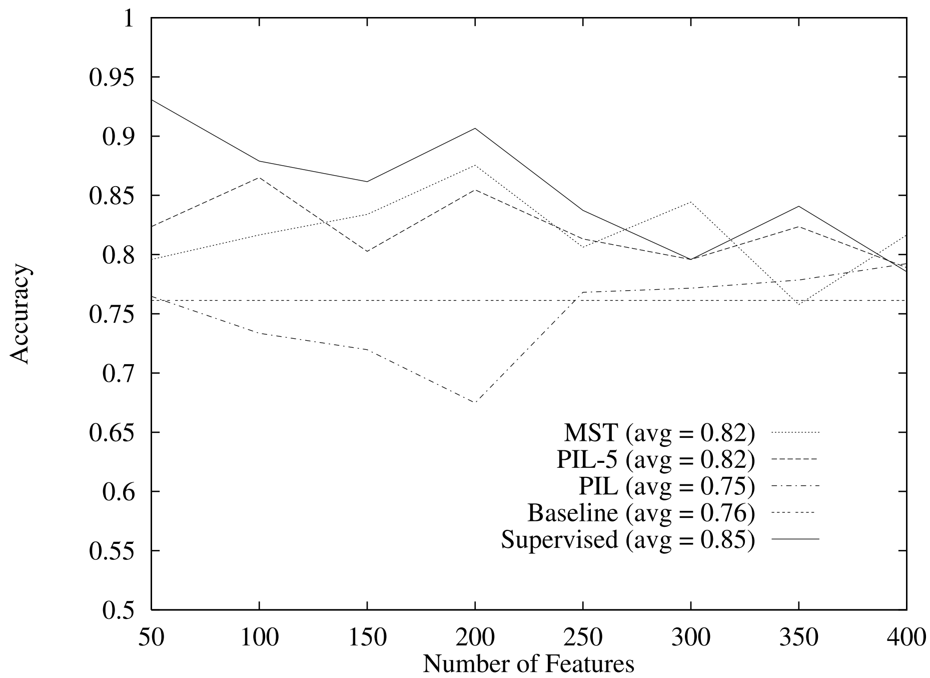

Figure 5.4: Results on Dataset 3.

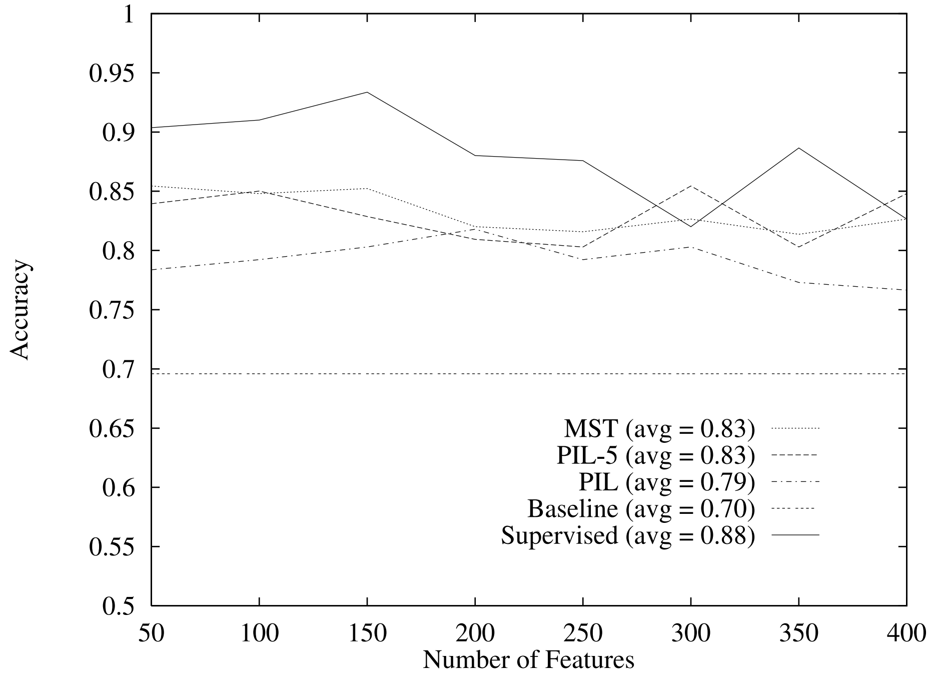

Figure 5.5: Results on Dataset 4.

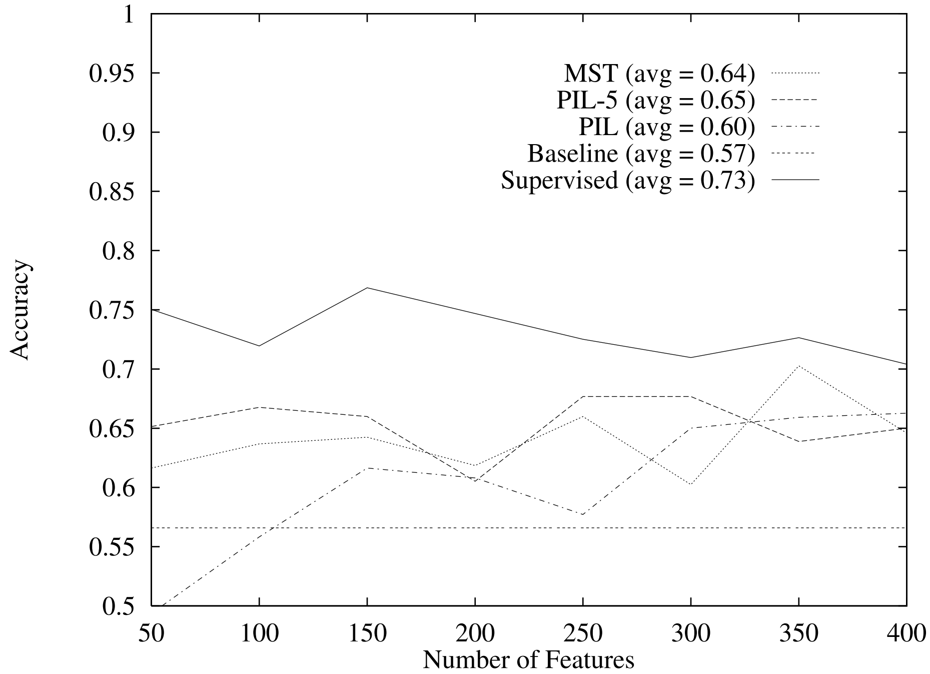

Figure 5.6: Results on Dataset 5.

original size. Also worth noting is that these methods provide consistent improvements, because they are in essence guiding the mixture model to capture the most significant feature interactions in the data without being hindered by the large variance of estimating all the parameters needed with the original data set. As to be expected, the *Supervised* method performs best overall, but shows that the unsupervised methods are generally performing quite well *without* having access to the class label information (which in practice would never be available).

The consistently poor performance of *PIL*, relative to the other feature selection methods, indicates that the approximation (used to measure the degree of influence each variable exerts on the rest of the domain) underlying this algorithm is not likely to be accurate. However, we see that *PIL-5* seems to exhibit more reasonable performance, even though it is based on a similar assumption. We believe that, since this variant of the algorithm only considers interactions between each feature and some limited number of other features, it does not cause the assumption on which it is based to be violated as wildly as in *PIL*. Furthermore, by making use of fewer features in this approximation, the detrimental effects of variance in parameter estimation can be controlled. This reliance on using fewer features to compute the interaction of each feature with the rest of the domain helps make *PIL-5* more roughly comparable in resulting accuracy to *MST*.

5.6 Conclusions

We have presented two novel algorithms for unsupervised feature selection based on theoretical considerations that appear to provide nice empirical improvements in using mixture models to cluster high-dimensional text data. Furthermore, our results provide even more evidence of the utility of using restricted dependency models in data fitting problems in high-dimensional spaces—a point we return to in Chapter 8.

While our initial results have been quite encouraging, a further analysis reveals some of the underlying limitations of using mixture models for text. In the experiments presented above, AutoClass was allowed to automatically select the number of clusters using the scoring criterion given in Section 3.2.2. In virtually every run,

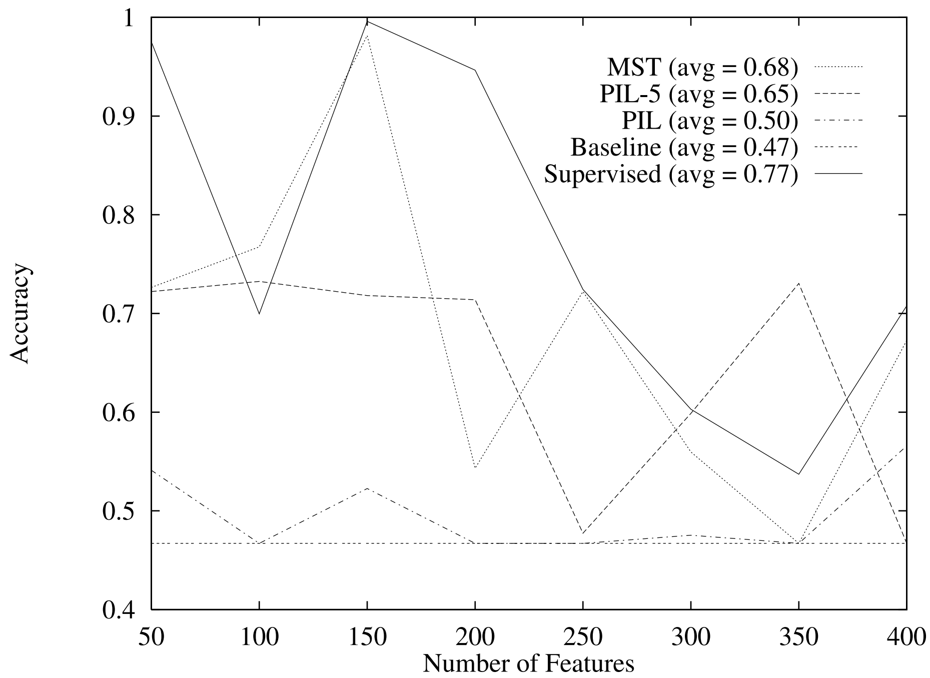

Figure 5.7: Results on Dataset 1 using three clusters.

AutoClass produced a model with between 20 and 25 clusters. This is quite remarkable given that the data sets used in this study were each composed of documents from three different topics. Still, it could be the case that AutoClass was discovering a finer level of topical granularity in these data sets.

To further test how well AutoClass could recover the topical partitioning of the data, we re-ran the experiments above, but in this case limited the algorithm to only construct models with three mixture components in order to match the true number of topics in the data. Recalling that inducing a mixture modeling requires performing an optimization in a non-convex space, it is important to note that the AutoClass algorithm actually builds several models in each run (we always built five models for each of the runs reported here) and selects the best one according to its model scoring criterion. In this way, it attempts to attenuate the effect of converging to a poor local maxima during the optimization. The results of these experiments are given in Figures 5.7 through 5.11.

In these results, we find that our feature selection methods are still effective at

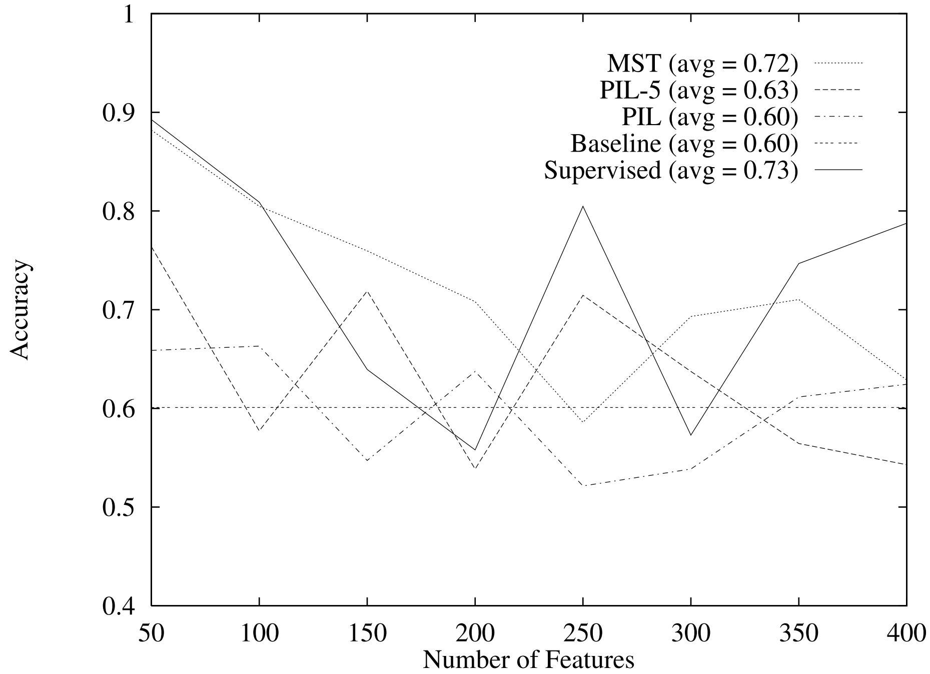

Figure 5.8: Results on Dataset 2 using three clusters.

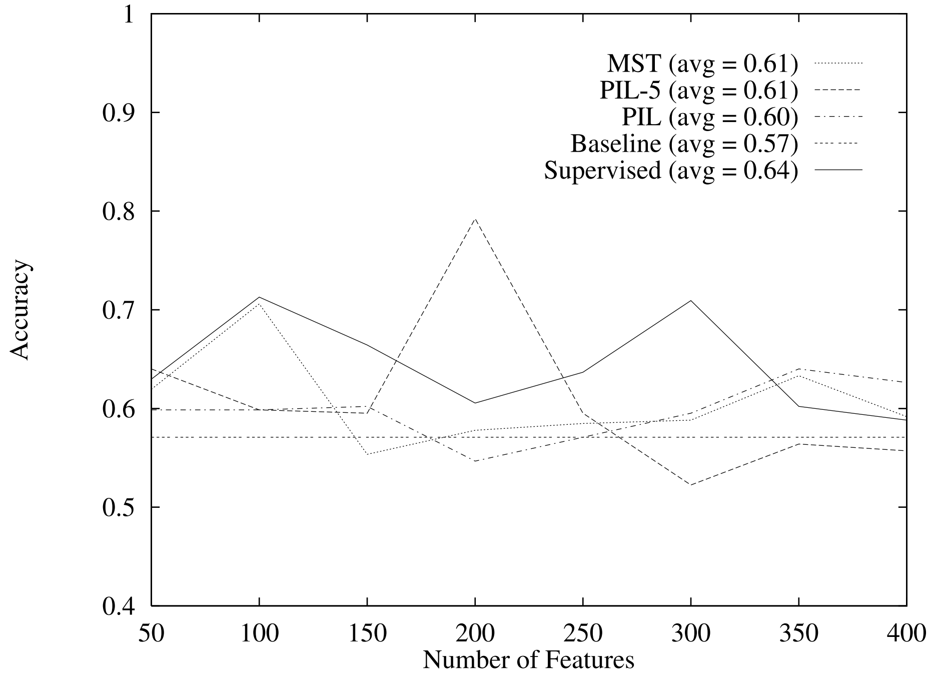

Figure 5.9: Results on Dataset 3 using three clusters.

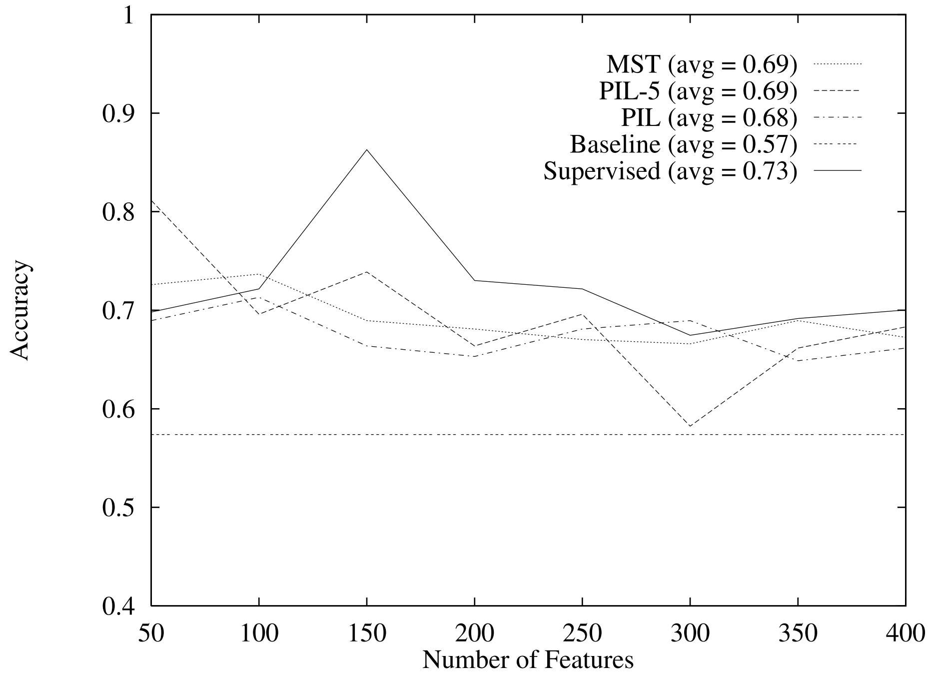

Figure 5.10: Results on Dataset 4 using three clusters.

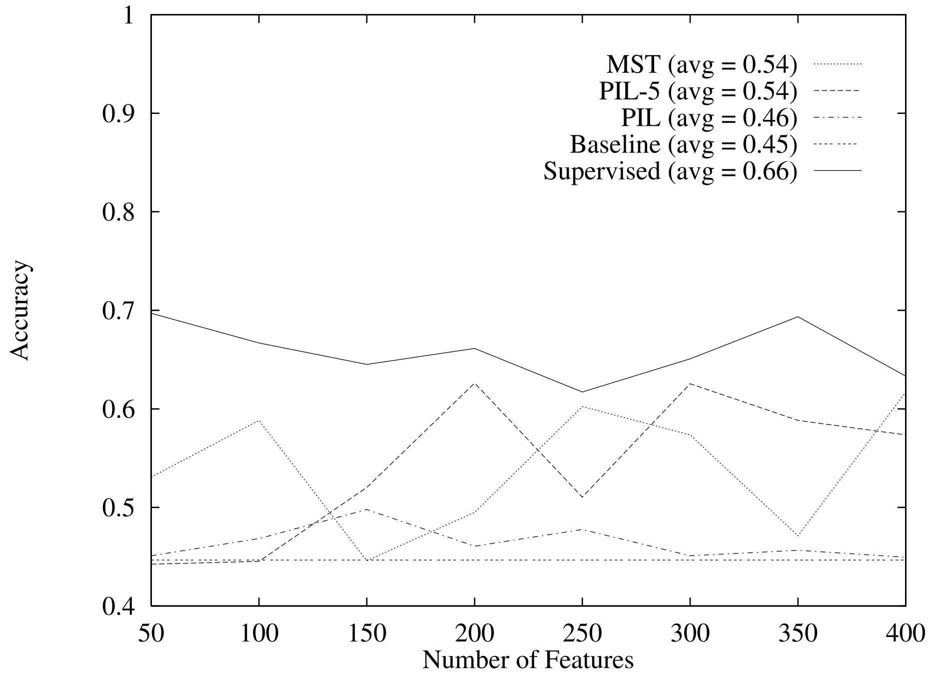

Figure 5.11: Results on Dataset 5 using three clusters.

producing better models than the original data. In fact, averaging over the different sized feature sets, every one of our feature selection methods outperformed the Baseline on every data set! Still, in contrast to the efficacy of our feature selection algorithms, several facts of applying mixture models to text domains become painfully clear. Foremost, the high variability of the accuracy results shows that there are many local maxima in the search space over which the mixture model is being optimized. As a result, it is often possible to produce a poor model, and consistency is hard to guarantee (even though we may induce several models from different random initial starting points and pick the best one according to some scoring criterion). Second, when the accuracy results obtained via mixture modeling are considered on an absolute scale, they still leave something to be desired. In essence, we would like to generate a clustering which better captures the topics we believe exist in corpus. This leaves us with the conclusion that, while feature selection appears to improve the results of mixture modeling, we would really like to have a better overall method for clustering, if possible. We turn our attention to this topic in the next chapter.

Chapter 6

A New Model for Document Clustering

6.1 Introduction

As explained in Chapter 1, the capability to effectively cluster documents of similar content can aid in both the retrieval of information and its presentation to the user (e.g., in a system like SONIA). Recall that early work in information retrieval stressed the use of clustering as a means of improving the ability to find documents relevant to a query [167] [144]. This work was based on the *Cluster Hypothesis* [166], which states that "closely associated documents tend to be relevant to the same requests." With this as a working assumption, document collections could be clustered a priori, and then new queries could simply be matched against clusters rather than against each document individually. Such cluster-based matching could speed the retrieval process and possibly find relevant documents that do not explicitly contain the words in the user's query.

More recently, applications of document clustering such as Scatter/Gather [39, 71] have been used to enable entire collections and query retrieval results to be browsed more easily. Similarly, our goal in SONIA is to use document clustering as a means of helping users quickly organize a collection of documents (e.g., the results of query retrieval). Moreover, by grouping related documents together, users can more readily browse the topics inherent in a corpus.

The success of systems such as SONIA can be significantly impacted by the effectiveness of the clustering methods employed. In our very preliminary work, for example, we found that using different clustering methods often led to different results with respect to how a collection of documents was characterized at a topical level. Indeed, the description of Scatter/Gather is very specific about the clustering methods used (albeit much more for reasons of efficiency than efficacy) reflecting the years of comparative work in the IR community. Such comparative analyses of different methods and document representations for clustering continue today [149].

While empirical work in document clustering has advanced the state of the art in performance, no equivalent advancement in theoretical analysis explains why the methods arrived at through experimentation work as well as they do. In fact, as seen in the previous chapter, many of the theoretically well-understood methods for clustering, such as mixture modeling, do not perform as well as we would hope on textual domains. In this chapter, we seek to provide an analysis of document clustering with the tools of probability theory. In this way, we can formalize the assumptions and models used in existing document clustering methods. Our objective is to gain new insights into the effectiveness of current clustering algorithms as well as to open the door to improvements. Uncovering the explicit distributional assumptions made in many text clustering algorithms has prompted us to investigate issues such as the treatment of evidence and different approaches to density estimation. From these insights, we have developed a new probability-based score for measuring document similarity. We provide empirical results showing that this score outperforms traditional IR methods for the text clustering task.

In general terms, the clustering problem consists of finding groups of data points that possess strong internal similarities. The problem is not formalized until we define what is meant by similarity. In practice, this formalization involves two separate issues: first, how one should measure similarity between data samples, and second, how one should evaluate a partitioning of a set of samples into clusters. We note, for clarity, that mixture modeling (as exemplified by AutoClass) also fits our general definition of clustering, even through the distance measures may not be explicitly defined. Here, however, we propose an explicit score for measuring similarity between documents derived from the probabilistic overlap in the words contained in each document.

In the context of clustering, and more generally throughout IR, a commonly used measure of similarity is obtained by representing documents as normalized vectors (as explained in Chapter 2) and then computing the inner product to find the cosine of the angle between the vectors. This measure of similarity is generally referred to as the *cosine coefficient* [144]. To compute this measure, documents are represented as vectors containing the normalized frequency counts (as opposed to the Boolean indicators) of the words in them. Intuitively, this measure tries to capture the degree of word overlap between two documents.

On similar grounds, in Section 6.2 we investigate a probabilistic function for document overlap that scores the expectation of the same words appearing in two documents. This score prompts the investigation of different smoothing methods for estimating the probability of a word appearing in a document. Section 6.3 describes the clustering algorithms we use in conjunction with this similarity score.

As our empirical evaluation in Section 6.4 shows, different smoothing methods may be more or less effective, depending on the degree of separability between the clusters. Furthermore, we point out some correspondencies between the document similarity measure derived in our framework and other probabilistic measures from the pattern recognition community, which have been derived in very different contexts. We also show that the widely used cosine coefficient can be associated with a particular form of probabilistic smoothing in our framework. Moreover, this analysis reveals a scaling factor, given by the inverse of the probability of a word appearing in the corpus, that, when combined with our similarity score, yields a clustering method that outperforms those based on the cosine coefficient and TFIDF weighting [145] in our experiments.

For completeness, in Section 6.5 we directly compare our new results with those obtained with mixture modeling using AutoClass. As expected, our new clustering method produces much better results, showing that it is quite suitable for incorporation into the SONIA system. Finally, we conclude in Section 6.6 with some general observations and directions for future work.

6.2 Probabilistic Document Overlap

To formalize the problem of document clustering, we first need to explicitly define a notion of similarity between documents. The similarity function that we will use for clustering will be based on establishing the degree of overlap between pairs of documents. To this end, we will assume that each document imposes a multinomial distribution over the set of words in the corpus. Each document doc_i is associated with an n-dimensional feature vector d_i, where the value of the jth component of the vector is the number of times the word corresponding to this component appears in the document. Thus, this vector representation of documents provides the sufficient statistics for computing the *expected overlap* between any given pair of documents (defined below).

Let doc_i and doc_j be two documents in a corpus D. We will then compute the expected overlap between doc_i and doc_j in terms of the corresponding vectors d_i and d_j. We denote this expected overlap measure as $\mathrm{EO}(d_i, d_j, D)$ and compute it as follows:

$$\sum_{w \in d_i \cap d_j} \mathrm{P}(Y_i = w \mid d_i, M) \cdot \mathrm{P}(Y_j = w \mid d_j, M) \ , \tag{6.1}$$

where $Y_i = w$ denotes the event that a word randomly selected from document doc_i is equal to w. M, the model, includes the sufficient statistics about the corpus D, including the total number of times each word appears in the corpus.

Eq. 6.1 is intuitively appealing. It says that the overlap between two documents i and j can be computed by estimating the probability that each word appears in each document, and then multiplying these results. As will be seen shortly, the way the probability of word appearance is estimated will greatly influence the results of clustering. We focus on the different ways of estimating this probability from the statistics in each vector d_i and M, as well as the relationship of this equation to the cosine coefficient below. We first provide a derivation of Eq. 6.1.

6.2.1 Deriving the Probabilistic Overlap

Here we investigate one possible derivation of Eq. 6.1 and reveal its underlying assumptions. We start by defining the expected degree of overlap between two documents doc_i and doc_j in the corpus D, using the corresponding vectors of word statistics d_i and d_j. This definition is given by

$$\sum_{w \in W} \mathrm{P}(Y_i \in d_i, Y_j \in d_j, Y_i = w, Y_j = w \mid d_i, d_j, M) \ . \tag{6.2}$$

The event $Y_i \in d_i$ denotes whether the word assigned to Y_i appears in d_i (i.e., has a nonzero count). If the word assigned to Y_i does indeed appear in d_i, then the probability of this event is 1, otherwise this event has probability 0. Thus, the events $Y_i \in d_i$ and $Y_j \in d_j$ in Eq. 6.2 are simply indicator functions that limit the set of words that contribute to the sum to only those $w \in d_i \cap d_j$. This reduces the sum above to

$$\sum_{w \in d_i \cap d_j} \mathrm{P}(Y_i = w, Y_j = w \mid d_i, d_j, M) \ . \tag{6.3}$$

The probabilities of the events $Y_i = w$ and $Y_j = w$ are fully determined by the document vectors d_i and d_j (and the model M). Thus, these events are conditionally independent given d_i, d_j, and M, giving us

$$\sum_{w \in d_i \cap d_j} \mathrm{P}(Y_i = w \mid d_i, d_j, M) \cdot \mathrm{P}(Y_j = w \mid d_i, d_j, M) \ . \tag{6.4}$$

Furthermore, the event $Y_i = w$ is conditionally independent of d_j, given d_i, since $Y_i = w$ is fully determined by d_i. Likewise, the event $Y_j = w$ is conditionally independent of d_i, given d_j. As a result, Eq. 6.4 reduces to

$$\mathrm{P}(Y_i = w \mid d_i, M) \cdot \mathrm{P}(Y_j = w \mid d_j, M) \ , \tag{6.5}$$

which is equal to Eq. 6.1.

As was pointed out above, this derivation embodies a series of assumptions of probabilistic independence. We remark that, given the relation we establish in the next section, these assumptions are also present in the use of the cosine coefficient to compute similarity. Our analysis above merely makes these assumptions explicit, opening opportunities for further research on verifying or even finding ways to avoid making them. Such research, however, is not addressed here.

6.2.2 Probability Estimation and Smoothing

We now focus on estimating the term $\mathrm{P}(Y = w \mid d, M)$ in Eq. 6.1.[1] An initial approach is to take the maximum likelihood (ML) estimate for this probability:

$$\mathrm{P}_{ML}(Y = w \mid d, M) = \frac{\xi(w, d)}{\sum_{w' \in d} \xi(w', d)} , \tag{6.6}$$

where $\xi(w, d)$ is the number of times that word w appears in document *doc* (represented by the vector d).

This is bound to be a poor estimate, as some words that are "important" to the topic of a document may appear only a few times, whereas other "unindicative" terms may appear very often. Also, with shorter documents such as news clips or many Web pages, this estimate will be even more prone to word "spikes" (i.e., will have high variance).

In trying to control variance in estimating $\mathrm{P}(Y = w \mid d, M)$, it becomes critical to perform some type of smoothing. A simple smoothing technique that has been used in the context of computational linguistics [21] is to use the arithmetic mean (AM) of $\mathrm{P}_{ML}(Y = w \mid d, M)$ and the maximum likelihood estimate of the unconditional distribution, $\mathrm{P}_{ML}(Y = w \mid M)$, where

$$\mathrm{P}_{ML}(Y = w \mid M) = \frac{\sum_{d \in D} \xi(w, d)}{\sum_{d \in D} \sum_{w' \in d} \xi(w', d)} . \tag{6.7}$$

For the case of $\mathrm{P}(Y = w \mid M)$, the ML estimate is appropriate because this computation is an average over all documents in the entire corpus and is therefore likely to

[1] We drop the subscript previously used with Y for the sake of readability.

attenuate any word spikes that may appear in a single document. Formally, arithmetic mean smoothing yields

$$\mathrm{P}_{AM}(Y = w \mid d, M) = \frac{1}{2}\mathrm{P}_{ML}(Y = w \mid d, M) + \frac{1}{2}\mathrm{P}_{ML}(Y = w \mid M)\ . \tag{6.8}$$

Note that, for simplicity, we compute the unweighted mean using coefficients of $\frac{1}{2}$. However, methods for fitting these coefficients to obtain improved results in language modeling [24] and text classification [116] have been explored.

A novel form of smoothing that we introduce involves the taking the *geometric* mean (GM) of these two ML distributions[2]:

$$\mathrm{P}_{GM}(Y = w \mid d, M) = \mathrm{P}_{ML}(Y = w \mid d, M)^{\frac{1}{2}} \cdot \mathrm{P}_{ML}(Y = w \mid M)^{\frac{1}{2}}\ . \tag{6.9}$$

The GM estimate in Eq. 6.9 does not define a true probability distribution because it will generally not sum to 1. We thus introduce a true probability distribution based on the geometric mean, by simply adding a normalization factor. This gives us the following *normalized* geometric mean (NGM) estimate:

$$\mathrm{P}_{NGM}(Y = w \mid d, M) = \frac{\mathrm{P}_{ML}(Y = w \mid d, M)^{\frac{1}{2}} \cdot \mathrm{P}_{ML}(Y = w \mid M)^{\frac{1}{2}}}{\sum_{w' \in W} \mathrm{P}_{ML}(Y = w' \mid d, M)^{\frac{1}{2}} \cdot \mathrm{P}_{ML}(Y = w' \mid M)^{\frac{1}{2}}}\ . \tag{6.10}$$

We continue to pursue the unnormalized GM formulation further, since it is related to the computation of similarity between documents using the normalized vector dot product, also known as the "cosine coefficient." Moreover, it is also closely related to other similarity scores developed in the pattern recognition community [12].

Here, we examine the cosine coefficient in more detail. Consider the similarity score of two documents, doc_i and doc_j, computed by using the cosine represented by Eq. 6.11:

$$\sum_{w \in W} \frac{\xi(w, d_i)}{(\sum_{w' \in d_i} \xi(w', d_i)^2)^{\frac{1}{2}}} \cdot \frac{\xi(w, d_j)}{(\sum_{w' \in d_j} \xi(w', d_j)^2)^{\frac{1}{2}}}\ . \tag{6.11}$$

[2]This is also equivalent to taking the arithmetic mean in the log space: $\log \mathrm{P}_{GM}(Y = w \mid d, M) = \frac{1}{2}\log \mathrm{P}_{ML}(Y = w \mid d, M) + \frac{1}{2}\log \mathrm{P}_{ML}(X = w \mid M)$.

Note, however, that when the cosine similarity score is used in information retrieval and clustering, the raw frequency scores often are not actually used as the features in a document vector. Rather, these frequencies are attenuated by a monotone shrinkage factor such as the log or square root (as in Eqs. 2.1 or 2.2, respectively). It has been reported that for the document clustering task, using the square root generally appears to give better performance than using the log [39]. Incorporating this factor into Eq. 6.11 yields

$$\sum_{w\in W} \frac{\xi(w,d_i)^{\frac{1}{2}}}{(\sum_{w'\in d_i}(\xi(w',d_i)^{\frac{1}{2}})^2)^{\frac{1}{2}}} \cdot \frac{\xi(w,d_j)^{\frac{1}{2}}}{(\sum_{w'\in d_j}(\xi(w',d_j)^{\frac{1}{2}})^2)^{\frac{1}{2}}} = \sum_{w\in W} (\frac{\xi(w,d_i)}{\sum_{w'\in d_i}\xi(w',d_i)})^{\frac{1}{2}} \cdot (\frac{\xi(w,d_j)}{\sum_{w'\in d_j}\xi(w',d_j)})^{\frac{1}{2}} \; . \quad (6.12)$$

First, we point out that Eq. 6.12 is equivalent to the expected overlap measure using the square roots of two maximum likelihood estimates, and can be rewritten as

$$\sum_{w\in W} \mathrm{P}_{ML}(Y_i = w \mid d_i, M)^{\frac{1}{2}} \cdot \mathrm{P}_{ML}(Y_j = w \mid d_j, M)^{\frac{1}{2}} \; , \quad (6.13)$$

Moreover, we have recently discovered that the measure in Eq.6.13 is exactly equivalent to the Bhattacharyya distance [12], which has been used as a measure of similarity between two probability distributions by researchers in the pattern recognition community. Thus we believe that the use of geometric factors (as evidenced by the use of the square root) may be a fruitful venue in comparing the multinomial distributions over words induced by documents. A further discussion of the Bhattacharyya distance is given by Kailath [86] who points out many of the similarities and differences between this measure and the relative entropy measure (given in Eq. 5.1), which is more commonly used to compare probability distributions in an information theoretic sense. Kailath's work also relates the Bhattacharyya distance to Fisher's measure of information [99] which is frequently used in statistical classification. Very recent work by Jaakkola and Haussler [80] has in turn used the Fisher information as a foundation for the derivation of kernel functions for classification. In the case of multinomial distributions, as we have here, these kernel functions have a very similar functional form (i.e., vector dot product over the parameters of the multinomial distribution) to our expected overlap measure. Hence, many connections exist between our proposed overlap measure (using various parameter estimation methods) and work in other communities. Investigating these connections may serve as a basis for future work. However, this line of investigation is tangential to the remainder of our work in clustering, and we do not pursue it further here.

Returning to our derivation of the overlap measure, we note that the sum in Eq. 6.12 can be reduced to include only those words $w \in d_i \cap d_j$, since any words not in both documents will have $\xi(w,d) = 0$ for at least one of the documents and will not influence the sum. This gives us

$$\sum_{w \in d_i \cap d_j} \left(\frac{\xi(w,d_i)}{\sum_{w' \in d_i} \xi(w',d_i)}\right)^{\frac{1}{2}} \cdot \left(\frac{\xi(w,d_j)}{\sum_{w' \in d_j} \xi(w',d_j)}\right)^{\frac{1}{2}} . \tag{6.14}$$

Finally, if we cast Eq. 6.14 in terms of the unnormalized GM estimate defined above, we obtain

$$\sum_{w \in d_i \cap d_j} \frac{\mathrm{P}_{GM}(Y_i = w \mid d_i, M) \cdot \mathrm{P}_{GM}(Y_j = w \mid d_j, M)}{\mathrm{P}_{ML}(Y = w \mid M)} . \tag{6.15}$$

Thus, the cosine similarity metric with square root dampening that has found empirical success in the IR community (and is closely tied to other similarity measures used in pattern recognition) is actually utilizing a form of geometric smoothing to account for the high variability in word appearances. Furthermore, casting the cosine in terms of the GM estimate uncovers a scaling factor for the axes of the word space (i.e., the denominator in Eq. 6.15). Intuitively, this scaling makes sense, since it incorporates additional knowledge in the form of the frequency of word usage in the corpus to be clustered. In our experiments below we evaluate the expected overlap given by Eq. 6.1 using the various estimation and smoothing proposals introduced in this section, plus a variant that incorporates the scaling factor in Eq. 6.15. As will be seen, the best results are obtained by using the NGM of Eq. 6.10 augmented with the scaling factor from Eq. 6.15.

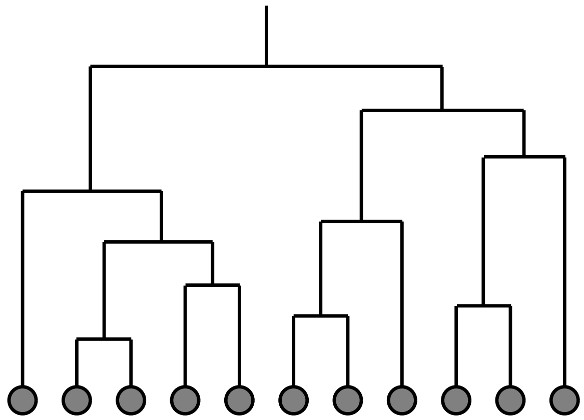

Figure 6.1: Example of a dendogram generated by hierarchical agglomerative clustering.

6.3 Clustering Algorithms

Having defined a similarity score for documents, we now turn to the problem of the actual document clustering algorithms. While a number of methods for clustering exist (some of which are discussed in Chapter 3), the two most widely applied to text domains are *hierarchical agglomerative clustering* (HAC) and *iterative clustering* techniques [131]. Both of these methods rely on the definition of a similarity score between pairs of documents. For the sake of generality, we will refer to this similarity score as $\mathrm{Sim}(doc, doc')$ and will subsequently instantiate it with our measure of probabilistic overlap, using different probability estimation methods.

6.3.1 Hierarchical Agglomerative Clustering

The most common clustering method employed in the information retrieval community over the past decade is hierarchical agglomerative clustering (HAC) [51]. This family of methods begins by placing each document into a distinct cluster. Pairwise similarities between all such clusters are computed, and the two closest clusters are then merged into a new cluster. This process of computing pairwise similarities and merging the closest two clusters is repeatedly applied, generating a *dendogram* structure that contains only one cluster (encompassing all the data) at its root. A sample dendogram is shown in Figure 6.1. By selecting an appropriate level of granularity in this dendogram, we can generate a partitioning into as many clusters as desired. Criteria such as a minimum number of documents per cluster are often used to prevent outlier documents from being considered a separate cluster. In our experiments we heuristically set this minimum cluster size at 10 documents.

Depending on how the similarity of a document to a cluster is defined, we can obtain different "flavors" of HAC; the most common are the *single link*, *complete link*, and *group average* methods. Previous work in IR [172] has pointed out that the group average method generally produces superior results, so we concentrate on this method here.

The group average method defines the similarity between a document *doc* and a cluster C as the average of the pairwise similarities between *doc* and each of the documents in C:

$$\mathrm{Sim}(doc, C) = \sum_{doc' \in C} \frac{1}{|C|} \mathrm{Sim}(doc, doc') \ . \tag{6.16}$$

Analogously, the similarity between two clusters C and C' is given as the average of the pairwise similarities between each document in C with each document in C'. More formally,

$$\begin{aligned} \mathrm{Sim}(C, C') &= \sum_{doc \in C, doc' \in C'} \frac{1}{|C| \cdot |C'|} \mathrm{Sim}(doc, doc') & (6.17) \\ &= \sum_{doc \in C} \frac{1}{|C|} \mathrm{Sim}(doc, C') \ . & (6.18) \end{aligned}$$

A simple probabilistic interpretation of the group average method is that each document in a cluster is an equally likely representative of that cluster. This is evident in the $\frac{1}{|C|}$ weighting given to each term in the sum in Eq. 6.16. Note that we can obtain many variations of HAC by replacing the term $\frac{1}{|C|}$ with alternate distributions over the "weight" of documents in a cluster (e.g., a Gaussian based on a document's distance from the cluster centroid).

6.3.2 Iterative Clustering

Iterative clustering techniques, also referred to as *reallocation* methods, attempt to optimize a given clustering by repeatedly reassigning documents to the cluster to which they are most similar. The general form for such algorithms, given a specification of the number of clusters k, is as follows:

```
1.  Initialize the k clusters[3]
2.  For each document doc in the corpus
    2.1.  Compute the similarity of doc to each cluster
3.  For each document doc in the corpus
    3.1.  Assign doc to the cluster most similar to it
4.  Goto 2, unless some convergence criterion is satisfied
```

As in the case of HAC, we define the similarity of a document to a cluster by the group average similarity. Our exit criterion in Step 4 can be met by simply running the algorithm for 10 iterations (although we observed that often far fewer were needed for convergence.)

It is worth noting that many well-known clustering algorithms, such as k-means [98], are instances of iterative clustering techniques. Moreover, such techniques (using appropriate similarity measures) can be viewed as *hard* assignment variants of probabilistic clustering techniques, such as AutoClass, that seek to produce models maximizing the likelihood of the data [46, 87, 117]. Unfortunately, since the optimization performed by iterative clustering is in a non-convex space, the initialization in Step 1 will affect the convergence point of the algorithm. We experimented using various runs with random initial clusters, and with using HAC as a method to find an initial clustering. The results of the former were often comparable and in some cases worse than the latter. Thus, we report only on the experiments where HAC determined the initial set of clusters.

[3] The random assignment of documents to clusters is one simple, commonly used method of initialization.

Dataset	U-ML	U-AM	U-NGM	S-ML	S-AM	S-NGM	Cos	TFIDF
D1	0.14	0.09	0.22	0.27	0.30	0.34	0.41	0.26
D2	0.16	0.11	0.26	0.35	0.38	0.43	0.47	0.28
D3	0.25	0.22	0.42	0.35	0.42	0.49	0.54	0.35
D4	0.17	0.11	0.31	0.34	0.38	0.48	0.52	0.32
D5	0.26	0.16	0.40	0.37	0.41	0.48	0.63	0.47

Table 6.1: Ratios of average inter-label to intra-label similarity.

6.4 Results

The objective of the experiments we describe in this section is to measure the efficacy of our proposed overlap measure, as well as to test the different estimation schemes for the computation of the expected overlap between documents. We are also interested in evaluating the effect of axis scaling on the expected overlap measure revealed from the derivation of the cosine coefficient. As will be seen below, the scaled NGM score of overlap performs better (in some case dramatically better) than any other score we tested, including the cosine coefficient and the TFIDF weighting method commonly used in IR.

To evaluate these clustering methods, we continue to use the evaluation criteria introduced in the previous chapter. Since we are interested in comparing the various probability estimation schemes as well as the absolute accuracy of our method, we a priori specified the number of clusters to be fitted in all the clustering runs to be the number of known class labels in the data. Recall, however, that the clustering algorithm is given no information about the true label of each document.

In a more general setting, the automatic determination of a good number of clusters is an open question. Nevertheless, we believe that in an interactive context, such as that offered by SONIA, it is easy for users to generate different numbers of clusters with the system and simply keep the one that best fits their specific organizational needs. In essence, the end-user is providing their own subjective model selection criterion in this task. Hence, the question of automatically determining the number of clusters (i.e., selecting the proper model) may be deferred to future work without seriously impinging on the development of SONIA.

The experiments reported here were conducted on the same subsets of the Reuters-22173 dataset used in the previous chapter. They are described in Table 5.1. Note, that we did not apply the feature selection techniques described in Chapter 5 to these datasets for the experiments conducted here. We did, however, use the same standard stop word elimination and Zipf's-Law-based feature selection described previously.

We empirically evaluated our measure of probabilistic overlap, using a number of different estimation schemes. First, we computed document overlap using the ML, AM, and GM estimates for $\mathrm{P}(Y = w \mid d, M)$. In these cases we did not scale the axes of the word space, so these computations are denoted "Unscaled" (U-). We then modified the computation of document overlap to include a scaling factor based on the marginal probability of word appearance, yielding

$$\sum_{w \in d_i \cap d_j} \frac{\mathrm{P}(Y_i = w \mid d_i, M) \cdot \mathrm{P}(Y_j = w \mid d_j, M)}{\mathrm{P}_{ML}(Y = w \mid M)} \ . \tag{6.19}$$

We identify these runs as "Scaled" (S-). For comparison, we also performed clustering using the cosine coefficient (with square root dampening) as a similarity score (as in Eq. 6.15). Also, recognizing the extensive use of TFIDF weighting in IR as an alternate means of term scaling, we also used this weighting scheme, in conjunction with the cosine rule (without square root dampening), as yet another similarity score for comparison. For our TFIDF weighting we employed the commonly used scheme given by Eq. 2.5.

Seeking to better characterize the datasets in our study according to the difficulty of recovering the underlying class structure, we also measured the ratio of the average inter-label similarity with the average intra-label similarity using the various similarity measures. These values, shown in Table 6.1, indicate the relative difficulty we would expect each measure of similarity to have with each dataset. An increase in these values indicates that documents within a class appear more and more similar to documents outside the class, thus making the recovery of the true class structure much more difficult. From these values we find that datasets D1, D2, and D3 are clearly in order of increasing difficulty for *all* the similarity measures. Datasets D4 and D5 show more relative variability. However, D5 is the most difficult dataset

Dataset	U-ML	U-AM	U-NGM	S-ML	S-AM	S-NGM	Cos	TFIDF
D1	95.7%	95.7%	95.1%	94.7%	95.3%	97.7%	**99.0%**	96.1%
D2	93.1%	95.9%	**98.9%**	86.7%	91.0%	95.3%	94.6%	90.6%
D3	68.2%	77.5%	76.8%	88.2%	83.7%	**96.9%**	79.2%	87.5%
D4	76.0%	52.7%	76.2%	90.6%	89.5%	**94.6%**	44.1%	57.6%
D5	74.2%	**78.5%**	44.2%	64.8%	72.8%	73.5%	49.7%	49.2%

Table 6.2: Accuracy percentages from hierarchical agglomerative clustering.

Dataset	U-ML	U-AM	U-NGM	S-ML	S-AM	S-NGM	Cos	TFIDF
D1	97.9%	99.0%	98.1%	99.2%	99.2%	**99.4%**	**99.4%**	**99.4%**
D2	97.4%	**99.1%**	98.5%	92.5%	97.0%	98.5%	97.0%	95.7%
D3	68.5%	79.6%	79.6%	95.2%	94.5%	**95.8%**	76.7%	92.4%
D4	90.0%	90.5%	88.1%	93.4%	93.4%	**95.9%**	80.9%	60.0%
D5	75.3%	68.0%	68.4%	70.8%	75.1%	**79.7%**	77.6%	52.7%

Table 6.3: Accuracy percentages from iterative clustering using HAC seeding.

when averaged over all the methods. These relative degrees of cluster identification difficulty are clearly reflected in the results of our experiments.

The accuracy percentages for clustering using HAC are given in Table 6.2, with the best results for each dataset boldfaced. Those for iterative clustering, using HAC as an initialization, are given in Table 6.3, and again the best results for each dataset are boldfaced. We note that applying the iterative optimization after performing HAC almost always leads to improved results, as seen in the increases in accuracy from Table 6.2 to Table 6.3. Hence, we focus our attention on Table 6.3.

Our first conclusion is that the use of axis scaling often improves the performance of the similarity measure using ML, AM, and NGM estimates. As a matter of fact, the accuracy is increased in 11 cases (often drastically), is decreased in 3 cases (only slightly), and remains unchanged in 1 case. This improvement reflects the obvious fact that words are not distributed uniformly in document collections, and taking advantage of this information can be of use in clustering.

Our second, and most important, conclusion highlights the utility of S-NGM as a similarity score. In general, the scaled probabilistic similarity measures using ML, AM, and NGM perform extremely well in comparison to both the cosine and TFIDF similarity scores, which are currently the state of the art in information retrieval. Most significantly, we draw the reader's attention to the S-NGM similarity score, which *always* produces an accuracy rate comparable to or much greater than that of either the cosine or the TFIDF methods! Noting that the cosine coefficient is equivalent to a scaled but unnormalized GM estimate, we see that the use of normalization to obtain true probabilities, as in the S-NGM case, not only can preserve more of the probabilistic flavor of our overlap measure, but also can have a significant beneficial impact on the empirical performance. Finally, in terms of absolute performance, we find that the accuracy figures obtained using S-NGM show that it is quite suitable for use in an end-user system like SONIA.

6.5 Comparison With Mixture Modeling

One of the original motivations for carrying out the work in this chapter stemmed from the poor results obtained previously in using probabilistic mixture modeling for document clustering. Thus, to further highlight the efficacy of our new method, we seek to compare it with probabilistic mixture modeling (using AutoClass). As explained in Chapter 5, using AutoClass requires that documents be represented as binary vectors (rather than word frequency counts). Then document clusters are obtained by fitting mixtures of independent binomial distributions over word appearances in the documents. Note that this Boolean representation has two immediate consequences: (1) it loses word frequency information and (2) it treats evidence about whether or not a word appears in a document in a symmetrical manner.

The loss of word frequency estimation may be remedied by the use of more complex statistical models (e.g., parametric distributions, such as bounded Gaussians or Poissons, over word frequencies) to fit the data. This approach, however, requires a commitment to a particular parametric model of word appearance. Our initial investigation along these lines, using bounded Gaussian distributions, indicates that this approach may not be promising. Still, as noted in Chapter 2, the loss of word

	HAC		HAC + Iter		AutoClass		
Dataset	S-NGM	Cos	S-NGM	Cos	S-NGM	Cos	Random
D1	97.7%	60.5%	**99.6%**	**99.6%**	97.9%	61.7%	46.7%
D2	92.3%	64.2%	**94.6%**	86.1%	91.0%	64.4%	60.1%
D3	70.9%	77.2%	70.9%	**84.4%**	64.7%	72.7%	57.1%
D4	76.0%	48.2%	**85.4%**	56.7%	70.6%	46.4%	57.4%
D5	**77.4%**	49.2%	77.3%	56.2%	72.2%	48.4%	44.7%

Table 6.4: Accuracy percentages using binary data.

frequency information alone may not be an especially important concern.

However, in the context of text clustering, the symmetrical treatment of evidence is more problematic. By "symmetrical treatment" we mean that word appearance and absence are given the same weight in a binomial distribution such as the one described above. One would expect, however, that the appearance of particular words in a text would be more indicative of a particular topic than the absence of some other word. Note that our probabilistic model of documents (which is based on a single multinomial) places much more importance on the information about the appearance of words than on their absence. Thus, the model matches our intuitions about word usage in text.

To test these arguments, we convert the datasets previously described to binary representations. The objective is to compare the two probabilistic models on fair grounds by removing the word frequency information. We then cluster this data, using the S-NGM and cosine similarity scores. (We do not compare the TFIDF similarity score here since it requires term frequency information to compute meaningfully). As before, we use both HAC alone and HAC followed by iterative clustering as the clustering methods. We also run AutoClass (which is a priori given the proper number of clusters to find), with initial clusters set with the results from HAC or via random seeding. To help alleviate the problems with bad initial conditions in the random case, we run AutoClass multiple times with different random initial clusters, and report the results for the best clustering chosen according to AutoClass's own model selection criterion. The results of these experiments are given in Table 6.4.

As expected, the lack of word frequency information generally hinders both S-NGM and cosine across both non-AutoClass clustering regimes. As we saw in the previous chapter, AutoClass with random initialization fails to find any real structure in any of the datasets. Furthermore, even when "reasonable" initial clusters are provided by HAC (using S-NGM and cosine), AutoClass outputs final clusters that are worse than the other methods. Thus, we believe that using our new model for document clustering, especially with the S-NGM estimate, can overcome the performance problems witnessed when applying mixture models to text.

As an aside, we note that while the intuition about asymmetry of evidence is useful for text clustering where categories must be discovered, it generally does not hold true for text classification tasks where the categories are known a priori. Recent work [115] has tried to address this issue more directly, arguing in favor of incorporating frequency information in classification. Nevertheless, it has also been observed empirically that applying such symmetric probabilistic models to text classification problems is still quite successful [110, 95]. We discuss the classification problem more fully in the next three chapters.

6.6 Conclusion

We have presented a probability-based measure for document similarity that is quite effective for clustering. We have also shown how the widely used cosine similarity coefficient can be captured as a particular form of probability estimation within our framework. Moreover, this formulation of the cosine coefficient has revealed a scaling factor that can be effectively integrated into our probabilistic framework and that yields results superior to those of traditional IR methods. Most importantly, the absolute performance of our new clustering model is good enough to make it a useful component of SONIA.

Although we do not pursue it here, we further point out that our similarity score can easily be extended to include more sophisticated notions of document overlap, based on equivalence classes of words (e.g., synonyms), phrases, or, in general, any function on groups of words in the corpus. In this way, our score can capture the full generality of *probabilistic indexing* [56] techniques used in other tasks (such as document retrieval). This extension can be performed by computing the expected document overlap as a sum over *multiple* multinomial distributions (one for each set of mutually exclusive functional events).

For example, say we wished to consider both single word and two-word phrases (i.e., bi-grams) in computing the overlap between documents. We could simply augment our current similarity score with another summation which computed the overlap in two-words phrases between documents. Thus, our final similarity score would incorporate two sums: one for the overlap in single words and another for the overlap in two-word phrases. The probabilities used in each sum would be normalized separately since the space of single words and two-word phrases can more easily be defined by two separate multinomial distributions, rather than trying to convolve them into one and somehow deal with the fact that the events are not mutually exclusive (i.e., a two-word phrase also constitutes two separate single word events). Moreover, each sum would be weighted in the overall similarity measure according to the relative influence we believe single words and two-word phrases to have in determining the topic of a document.

In this way we will be able to easily incorporate much more information than simple word frequencies into our similarity score. Moreover, the parameters defining the contributions of different words or functional characteristic of the documents to the overall similarity score in these cases can be learned directly from the data. Finally, we note that another advantage of a probability-based score is the possibility of easily fusing information coming from different modalities (such as video and audio) into similarity scores over multimedia domains. Indeed, we have already conducted some very preliminary, but promising, experiments in applying our similarity measure to problems in other domains such as video segmentation, using different estimation techniques as appropriate. This continuing work, however, is beyond the scope of the topic of text clustering discussed here.

Part III

Classification

Chapter 7

Feature Selection for Classification

7.1 Introduction

Having seen how clustering may be used to discover topics in a collection of documents, we now turn our attention to classifying documents into pre-defined categories. Note that such categories can either be the result of automatic clustering techniques or may be manually defined by a user. Moreover, as systems such as SONIA become more widely available, it is also likely that such categorization schemes may be frequently produced by some semi-automated combination of these two methods.

In the classification (i.e., *supervised learning*) task, we are given a training set of labeled fixed-length feature vectors, or instances, from which to induce a classification model. This model is then used to predict the class label for a set of unlabeled instances (i.e., uncategorized documents). Thus, the information about the class that is inherent in the features determines the accuracy of the model. Theoretically, having more features should give us more discriminating power. However, the real world, especially in the case of document classification, provides us with many reasons why this theoretical observation does not generally hold in practice.

It is well-known that the number of features can have a strong effect on the performance of a learning algorithm. First, the time requirements for an induction algorithm often grow dramatically with the number of features, rendering the algorithm impractical for problems with a large number of features. Since the computational complexity of such learning algorithms depends heavily on the number of features, it is often useful to reduce the feature set prior to constructing a classifier.

Furthermore, many learning algorithms, such as decision trees and, obviously, Bayesian classifiers, can be viewed as performing (a biased form of) estimation of the probability of the class label given a set of features. In domains such as text with a large number of features, this distribution is very complex and of high dimension. Unfortunately, in the real world, we are often faced with the problem of limited data from which to induce a model. This limited availability of data makes it very difficult to obtain good estimates of the many probabilistic parameters. Thus, the estimation of more parameters resulting from using more features in a classifier can cause the detrimental effects of increased parameter variance to outweigh potential gains that might be realized from the information contained in the additional features. This problem is best characterized by the *bias/variance tradeoff* [61].

Moreover, irrelevant and redundant features also cause problems in this context as they may confuse the learning algorithm by helping to obscure the distributions of the small set of truly relevant features for the task at hand. The inclusion of such irrelevant or redundant features in the training data can, thus, degrade the accuracy of a learning algorithm, causing it to use less than optimal features for classification.

In order to avoid over-fitting the model to the idiosyncrasies of the training data and thereby reduce the variance in the estimation of the model parameters, many algorithms employ the Occam's Razor [13] bias to build as simple a model as possible that still achieves some acceptable level of performance on the training data. This bias often leads us to prefer a small number of relatively predictive features over a very large number of features that, taken in the proper, but complex, combination, are entirely predictive of the class label.

If we reduce the set of features considered by the algorithm, we can therefore serve two purposes. We can considerably decrease the running time of the induction algorithm, and we can increase the accuracy of the resulting model. In light of this, a number of researchers have recently addressed the issue of feature subset selection in machine learning. As noted by John, Kohavi and Pfleger [84], this work is often divided along two lines: *filter* and *wrapper* models.

In the filter model, feature selection is performed as a preprocessing step to induction. Thus, the bias of the learning algorithm does not interact with the bias inherent in the feature selection algorithm. Two of the most well-known filter methods for feature selection are RELIEF [88] and FOCUS [5]. In RELIEF, a subset of features is not directly selected, but rather each feature is given a weighting indicating its level of relevance to the class label. RELIEF is therefore ineffective at removing redundant features since two predictive, but highly correlated, features are both likely to be highly weighted. The FOCUS algorithm conducts an exhaustive search of all feature subsets to determine the minimal set of features that can provide a consistent labeling of the training data. This consistency criterion makes FOCUS very sensitive to noise in the training data. Moreover, the exponential size of the power set of the features makes this algorithm impractical for domains with more than 30 features.

Another feature selection methodology which has recently received much attention is the wrapper model [84, 20, 104]. This model searches through the space of feature subsets using the estimated accuracy from an induction algorithm as the measure of goodness for a particular feature subset. Thus, the feature selection is being "wrapped around" an induction algorithm, so that the bias of the operators that define the search and that of the induction algorithm strongly interact. While these methods have encountered some success on induction tasks, they are often prohibitively expensive to run and can be intractable even for a few hundred features. Furthermore, the methods leave something to be desired in terms of theoretical justification. While an important aspect of feature selection is how well a method helps an induction algorithm in terms of accuracy measures, it is also important to understand how the induction problem in general is affected by feature selection.

We address here both theoretical and empirical aspects of feature selection with respect to the classification task. We describe a formal framework for understanding feature selection, based on ideas from Information Theory [35]. We then present an efficient implemented algorithm based on these theoretical intuitions. The algorithm overcomes many of the problems with existing methods: it has a sound theoretical foundation; it is effective in eliminating both irrelevant and redundant features; it is tolerant to inconsistencies in the training data; and, most importantly, it is a filter algorithm which does not incur the high computational cost of conducting a search through the space of feature subsets as in the wrapper methods, and is therefore efficient for domains containing several hundred or even thousands of features.

The formal framework for feature selection is presented in Section 7.2. Section 7.3 presents an algorithm for feature selection based on this framework. Empirical results for this algorithm are given in Sections 7.4 and 7.5. We conclude in Section 7.6 with a discussion of this work.

7.2 Theoretical Framework

Recall that a *classifier* is a procedure that takes as input a data instance $\boldsymbol{x}$ which is an assignment of values $x_1, \ldots, x_n$ to a set of features $\boldsymbol{X} = (X_1, \ldots, X_n)$. The classifier then predicts that the instance is a member of one of K possible classes $c_1, \ldots, c_K$. The classifier must make its decision based on the assignment $\boldsymbol{x}$ associated with an instance. Optimistically, the feature vector will fully determine the appropriate classification. However, this is rarely the case: we do not typically have access to enough features to make this a deterministic decision. Therefore, we use a probability distribution to model the classification function. For each assignment of values $\boldsymbol{x}$ to $\boldsymbol{X}$, we have a distribution $\mathrm{P}(C \mid \boldsymbol{X} = \boldsymbol{x})$ on the different possible classes, C. A learning algorithm implicitly uses the empirical frequencies observed in the training set — an approximation to the conditional distribution $\mathrm{P}(C \mid \boldsymbol{X})$ — to construct a classifier for the problem.

Let us consider the effect of feature space reduction on the distribution that characterizes the problem. Let $\boldsymbol{Y}$ be some subset of $\boldsymbol{X}$. Given a feature vector $\boldsymbol{x}$, we use $\boldsymbol{x}_{\boldsymbol{Y}}$ to denote the projection of $\boldsymbol{x}$ onto the variables in $\boldsymbol{Y}$. Consider a particular data instance characterized by $\boldsymbol{x}$. In the original distribution, this data instance induces the distribution $\mathrm{P}(C \mid \boldsymbol{X} = \boldsymbol{x})$. In the reduced feature space, the same instance induces the (possibly different distribution) $\mathrm{P}(C \mid \boldsymbol{Y} = \boldsymbol{x}_{\boldsymbol{Y}})$. Our goal is to select $\boldsymbol{Y}$ so that these two distributions are as close as possible. As our distance metric, we use the information-theoretic measure of relative entropy given in Eq. 5.1. Thus, we can view process this as selecting a set of features $\boldsymbol{Y}$ which causes us to lose the least amount of information in these distributions. While other measures of separability (notably *divergence*) have been suggested in the statistics community for feature selection [58], these measures are often aimed at selecting features to enhance the separability of the data and may have difficulty in very large dimensional spaces. Hence, they bring with them an inherent bias which may not be appropriate for particular induction algorithms. Our method seeks to eliminate non-informative features and thereby allow induction methods to employ their own bias in a much reduced feature space.

Recall that the relative entropy between two distributions μ and σ, denoted $D(\mu, \sigma)$, measures the extent of the "error" made by using σ as an approximation to μ. Thus, this measure is particularly suitable for our application, with $\mathrm{P}(C \mid \boldsymbol{x})$ in the role of the "more informed" distribution μ, and $\mathrm{P}(C \mid \boldsymbol{x}_{\boldsymbol{Y}})$ in the role of σ. In this case, the probability space Ω is the set of possible classifications $\{c_1, \ldots, c_K\}$. Therefore, we define

$$\delta_{\boldsymbol{Y}}(\boldsymbol{x}) = D(\mathrm{P}(C \mid \boldsymbol{x}), \mathrm{P}(C \mid \boldsymbol{x}_{\boldsymbol{Y}})) \ . \tag{7.1}$$

Of course, in order to have a measure which allows us to compare one feature set $\boldsymbol{Y}$ to another, we must integrate the values $\delta_{\boldsymbol{Y}}(\boldsymbol{x})$ for different feature vectors $\boldsymbol{x}$ into a single quantity. Naively, we might think to simply sum the relative entropy for the different feature vectors, or to consider the maximum relative entropy over all feature vectors. Neither of these ideas take into consideration the fact that some feature vectors are far more likely to occur than others, and that we might not mind making a larger mistake in certain rare cases. Therefore, we want to find a feature set $\boldsymbol{Y}$ for which $\Delta_{\boldsymbol{Y}} = \sum_{\boldsymbol{x}} \mathrm{P}(\boldsymbol{x}) \delta_{\boldsymbol{Y}}(\boldsymbol{x})$ is reasonably small.

Clearly, the feature set that minimizes this quantity is simply $\boldsymbol{X}$, since that maintains the exact distribution. This observation suggests that we use a backward elimination algorithm, where at each state we eliminate a feature X_i in a way that allows us to remain as close to this distribution as possible. Intuitively, we use a greedy algorithm where we eliminate the feature X_i which would cause us the smallest increase in Δ. That is, we have a current feature set $\boldsymbol{Y}$, initially set to $\boldsymbol{X}$. At each stage, we want to eliminate the feature X_i such that $\Delta_{(\boldsymbol{Y}-\{X_i\})}$ is as close as possible to $\Delta_{\boldsymbol{Y}}$.

Unfortunately, it is impractical to simply implement this idea as described, since the computation of $\Delta_{\boldsymbol{Y}}$ is exponential in the number of features in our domain. Furthermore, we cannot really compare our approximate distribution to the true conditional distribution $\mathrm{P}(C \mid \boldsymbol{X})$, since the true distribution is not available to us. Rather, we have a training set which provides us only a rough approximation to it. In those cases where we have a large number of features, the number of data instances in our training set corresponding to any particular assignment $\boldsymbol{x}$ will be very small. Therefore, as the number of features grows, our ability to use the training set to approximate this conditional distribution decreases (exponentially).

As we now show, we can utilize ideas from probabilistic reasoning to circumvent this problem (to some extent). Intuitively, features that cause a small increase in Δ are those that give us the least additional information beyond what we would obtain from the other features in $\boldsymbol{Y}$. We can capture this intuition via the formal notion of *conditional independence*, defined in Chapter 3.

Proposition 6 *Let $\boldsymbol{Y}$ be a subset of features in $\boldsymbol{X}$, and X_i be a feature in $\boldsymbol{Y}$. Then X_i is conditionally independent of C given $\boldsymbol{Y}' = \boldsymbol{Y} - \{X_i\}$ if and only if $\Delta_{\boldsymbol{Y}'} = \Delta_{\boldsymbol{Y}}$.*

Thus, we can eliminate a conditionally independent feature X_i from $\boldsymbol{Y}$ without increasing our distance from the desired distribution. Intuitively, removing a feature which is "almost" conditionally independent will not make our distance grow too large.

While it is also impractical to test for conditional independence given $\boldsymbol{Y}'$, this reformulation of the problem points the way to a solution. Intuitively, if all of the information in X_i is subsumed by the features in $\boldsymbol{Y}'$, it is almost certainly subsumed by some subset of the features in $\boldsymbol{Y}'$ as well. After all, it is very unlikely that *all* of these (usually very many) features are actually required.

Definition 2 *[125, p. 97] Let $\boldsymbol{M}$ be some set of features which does not contain X_i. We say that $\boldsymbol{M}$ is a* Markov blanket *for X_i if X_i is conditionally independent of $(\boldsymbol{X} \cup C) - \boldsymbol{M} - \{X_i\}$ given $\boldsymbol{M}$.*

It is easy to see that if $\boldsymbol{M}$ is a Markov blanket of X_i, then it is also the case that the class C is conditionally independent of the feature X_i given $\boldsymbol{M}$. Therefore we give the following corollary which is obtained directly from Prop. 6 and Def. 2:

Corollary 7 *Let $\boldsymbol{Y}$ be a subset of features in $\boldsymbol{X}$, and X_i be a feature in $\boldsymbol{Y}$. Assume that some subset $\boldsymbol{M}$ of $\boldsymbol{Y}$ is a Markov blanket of X_i. Then $\Delta_{\boldsymbol{Y}'} = \Delta_{\boldsymbol{Y}}$.*

However, the Markov blanket condition is stronger than requiring that X_i and C be conditionally independent, given $\boldsymbol{M}$. It requires that $\boldsymbol{M}$ subsume not only the information that X_i has about C, but also about all of the other features. While it might be difficult to find such a set $\boldsymbol{M}$, use of Markov blankets as the basis for feature elimination has a number of very desirable properties that we presently explore. Ideally, we would like to use the Markov blanket condition, if it is feasible to implement in a practical algorithm.

Intuitively, we want to remove features for which we find a Markov blanket within the set of remaining features. We now show that features judged as unnecessary based on this criterion remain unnecessary during the rest of the process. Assume, for example, that we remove a feature X_i based on a Markov blanket $\boldsymbol{M}$. At some later phase, we might remove some other feature $X_j \in \boldsymbol{M}$. In general, the removal of X_j might now render X_i relevant again; that is, if we were to add X_i back in, we might not be able to remove it again. As we now show, this is not the case.

Theorem 8 *Let $\boldsymbol{Y}$ be our current set of features, and assume that some (previously removed) feature $X_i \notin \boldsymbol{Y}$ has a Markov blanket within $\boldsymbol{Y}$. Let $X_j \in \boldsymbol{Y}$ be some feature which we are about to remove based on some Markov blanket within $\boldsymbol{Y}$. Then X_i also has a Markov blanket within $\boldsymbol{Y} - \{X_j\}$.*

Proof: The proof is based on the basic independence properties of probability distributions, as described in [125, p. 84]. We will use the notation $I(X, Y \mid Z)$ to denote the conditional independence of two variables or sets of variables X and Y given a set of variables Z. Let $\boldsymbol{M}_i \subseteq \boldsymbol{Y}$ be the Markov blanket of X_i (note that this is not necessarily the same Markov blanket which we used in order to remove X_i in the first place); let $\boldsymbol{M}_j \subseteq \boldsymbol{Y}$ be the Markov blanket which we are now using to remove

X_j. It is straightforward to show that if $\boldsymbol{M}_i$ does not contain X_j, then it remains a Markov blanket for X_i even after the removal of X_j from $\boldsymbol{Y}$. Therefore, consider the case where $X_j \in \boldsymbol{M}_i$ and define $\boldsymbol{M}_i = \boldsymbol{M}_i' \cup \{X_j\}$. We will show that $\boldsymbol{M}_i' \cup \boldsymbol{M}_j$ is a Markov blanket for X_i. Let $\boldsymbol{Z}$ denote $\boldsymbol{Y} - \{X_j\} - (\boldsymbol{M}_i' \cup \boldsymbol{M}_j)$. We need to show that $I(X_i, \boldsymbol{Z} \mid (\boldsymbol{M}_i' \cup \boldsymbol{M}_j))$. From the Markov blanket assumption for X_j and the Decomposition property, we have that $I(X_j, (\boldsymbol{Z} \cup \boldsymbol{M}_i') \mid \boldsymbol{M}_j)$. Using the Weak Union property, we obtain that $I(X_j, \boldsymbol{Z} \mid (\boldsymbol{M}_i' \cup \boldsymbol{M}_j))$. Similarly, we can derive that $I(X_i, (\boldsymbol{Z} \cup (\boldsymbol{M}_j - \boldsymbol{M}_i')) \mid \boldsymbol{M}_i' \cup \{X_j\})$, and therefore that $I(X_i, \boldsymbol{Z} \mid \boldsymbol{M}_i' \cup \boldsymbol{M}_j \cup \{X_j\})$. From these two facts, we can use the Contraction property to show the desired result. ∎

Thus, we conclude the Markov blanket criterion only removes attributes that are really unnecessary. As interesting is the fact that the converse of this statement is also true. Two types of attributes are generally perceived as being unnecessary: attributes that are irrelevant to the target concept, and attributes that are redundant given other attributes. The Markov blanket criterion captures both of these. Attributes that are irrelevant will be unconditionally independent of everything, so they will be removed based on a Markov blanket consisting of the empty set of features. Even if we have a set of attributes that are correlated only with each other, but are completely independent of the class variable, the Markov blanket criterion will remove them one by one: at each stage, the remaining irrelevant features will be used as a Markov blanket for the one we are trying to remove. If, on the other hand, we have a feature whose value is fully determined (or even probabilistically determined) by some set S, we will be able to remove it by using S as its Markov blanket.

It is interesting to compare our approach to another, seemingly very similar one, often used in the literature [153]. There, rather than starting with the full feature set and eliminating features, we begin with an empty set of features and add features one by one. Usually, the measure used to add features is *information gain*: we add to our current $\boldsymbol{Y}$ the feature X_j that maximizes the expected relative entropy between $\mathrm{P}(C \mid \boldsymbol{Y})$ and $\mathrm{P}(C \mid \boldsymbol{Y} \cup \{X_j\})$. It is easy to show that our idea of using a Markov blanket to estimate the relative entropy can also be applied in the case of forward selection. Therefore, it might seem that the two approaches are essentially

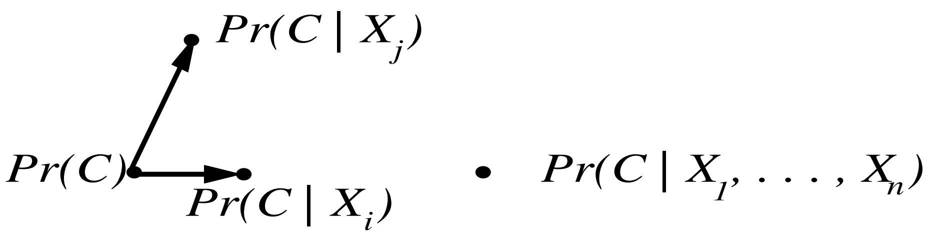

Figure 7.1: Forward vs. backward selection

minor variants on the same theme. We claim that this is not the case: Our formal framework provides us with the tools to compare forward selection and backward elimination, and justifies our choice of backward elimination.

Recall that our goal was to remain as close as possible to the "correct" conditional distribution $\mathrm{P}(C \mid \boldsymbol{X})$. By removing features that only take "small steps" away from this distribution, we can remain close to it. By contrast, the forward selection scheme starts out with the prior distribution $\mathrm{P}(C)$ given no features. It then tries to take "large steps" away from that distribution. If the goal of this process is to get as close as possible to the "right" distribution, the problem becomes clear. There is no guarantee that taking a large step away from initial distribution actually gets us closer to the goal distribution.

For example, as illustrated in Figure 7.1, adding X_j might let us take a much larger step than adding X_i, but the resulting distribution $\mathrm{P}(C \mid X_j)$ is actually further from the "right" distribution than $\mathrm{P}(C \mid X_i)$. As we show in Section 7.4, this behavior actually occurs on some of our data sets.

7.3 An Approximate Algorithm

Previously, we showed how we can eliminate a feature X_i from a candidate feature set $\boldsymbol{Y}$ by finding a Markov blanket $\boldsymbol{M}$ for X_i. Unfortunately, there might not be a full Markov blanket for a feature, but rather only an approximate one that subsumes the information content of the feature. Furthermore, finding either a true or an approximate Markov blanket might be computational infeasible. We now present a simple algorithm which provides a heuristic approach to dealing with this problem.

Broadly, our algorithm iteratively selects one candidate set $\boldsymbol{M}_i$ for each feature X_i, and uses a rough heuristic to estimate how close $\boldsymbol{M}_i$ is to being a Markov blanket for X_i; the feature X_i for which $\boldsymbol{M}_i$ is closest to being a Markov blanket is eliminated, and the algorithm repeats. Our intuition for constructing a candidate Markov blanket is as follows: Assume that X_i does, in fact, have a Markov blanket $\boldsymbol{M}_i$. We can think of X_i as *directly influencing* the features in $\boldsymbol{M}_i$. Therefore, these features will tend to be quite strongly correlated with X_i. Other features, on the other hand, are conditionally independent of X_i given $\boldsymbol{M}_i$. Thus, X_i influences them only indirectly, via $\boldsymbol{M}_i$. There is a well-known "folk-theorem" that probabilistic influence tends to attenuate over distance; that is, direct influence is typically stronger than indirect influence. (This has been shown formally and empirically in certain cases [45, 97].) Therefore, we heuristically choose, as an approximation to the Markov blanket, some set of K features which are strongly correlated with X_i. We tested a number of feature "correlation" metrics including statistical correlation, mutual information between features, class conditional mutual information and "pair-wise" relative entropy (described below). The latter measure provided the best initial results and was used in the experiments reported here.

We now want to figure out how close $\boldsymbol{M}_i$ is to being a Markov blanket for X_i. Unfortunately, evaluating the conditional independence expression in Definition 2 is typically very expensive. In fact, while the Markov blanket condition has many desirable properties outlined above, it is not feasible to directly implement in a practical algorithm. Rather, we use the intuitions afforded by this property to find a suitable approximation. First, we note that if $\boldsymbol{M}_i$ is really a Markov blanket for X_i, then

$$D(\mathrm{P}(C \mid \boldsymbol{M} = \boldsymbol{x}_{\boldsymbol{M}}, X_i = x_i), \mathrm{P}(C \mid \boldsymbol{M} = \boldsymbol{x}_{\boldsymbol{M}})) = 0$$

for any assignment of feature values $\boldsymbol{x}_{\boldsymbol{M}}$ and x_i to $\boldsymbol{M}$ and X_i, respectively.

While the Markov blanket condition requires that $\boldsymbol{M}$ subsume all information that a feature X_i has about both the class C and all other features, we really only care about the influence of a feature X_i on C. Thus, rather than trying to find a true Markov blanket, we can simply search for a set of features $\boldsymbol{M}$ which subsumes all the information which X_i has about C, without regard to the influence X_i may have on any other features. We therefore define the expected relative entropy:

$$\begin{aligned}\Delta_{\boldsymbol{Y}}(X_i \mid \boldsymbol{M}_i) = \sum_{\boldsymbol{x}_{M_i}, x_i} & \mathrm{P}(\boldsymbol{M}_i = \boldsymbol{x}_{\boldsymbol{M}_i}, X_i = x_i) \cdot \\ & D(\mathrm{P}(C \mid \boldsymbol{M} = \boldsymbol{x}_{\boldsymbol{M}}, X_i = x_i), \mathrm{P}(C \mid \boldsymbol{M} = \boldsymbol{x}_{\boldsymbol{M}}))\ . \end{aligned} \tag{7.2}$$

If we assume that $\mathrm{P}(C \mid \boldsymbol{Y})$ is a good approximation to $\mathrm{P}(C \mid \boldsymbol{X})$, then our goal is to stay close to the former. Note that this is an approximation since (among other reasons) relative entropy does not satisfy the triangle inequality, so that we cannot conclude anything about $D(\mathrm{P}(C \mid \boldsymbol{X}), \mathrm{P}(C \mid \boldsymbol{Y} - \{X_i\}))$ from $D(\mathrm{P}(C \mid \boldsymbol{X}), \mathrm{P}(C \mid \boldsymbol{Y}))$ and $D(\mathrm{P}(C \mid \boldsymbol{Y}), \mathrm{P}(C \mid \boldsymbol{Y} - \{X_i\}))$. If $\boldsymbol{M}_i$ is, in fact, a Markov blanket for X_i, then $\Delta_{\boldsymbol{Y}}(X_i \mid \boldsymbol{M}_i) = 0$. (This follows from the same techniques used in Corollary 7.) Hopefully, if it is an approximate Markov blanket, or at least subsumes most of the information that X_i has about C, then this value will be low.

It is also worth noting that our measure for feature selection given in Eq. 7.2 bears important similarities to other measures in information theory which we show below. Nevertheless, the derivation for the measure presented above is based on very different intuitions than the standard channel coding arguments on which much of information theory is based [35].

Theorem 9 *Let $MI(C; X_i \mid \boldsymbol{M}_i)$ denote the* mutual information *between the class C and feature X_i given $\boldsymbol{M}_i$. Then, $\Delta_{\boldsymbol{Y}}(X_i \mid \boldsymbol{M}_i) = MI(C; X_i \mid \boldsymbol{M}_i)$*

Proof:

$$\begin{aligned}
\Delta_{\boldsymbol{Y}}(X_i \mid \boldsymbol{M}_i) &= \sum_{\boldsymbol{x}_{M_i}, x_i} \mathrm{P}(\boldsymbol{x}_{\boldsymbol{M}_i}, x_i) \cdot D(\mathrm{P}(C \mid \boldsymbol{x}_{\boldsymbol{M}_i}, x_i), \mathrm{P}(C \mid \boldsymbol{x}_{\boldsymbol{M}_i})) && (7.3)\\
&= \sum_{\boldsymbol{x}_{M_i}, x_i} \mathrm{P}(\boldsymbol{x}_{\boldsymbol{M}_i}, x_i) \cdot \sum_c \mathrm{P}(c \mid \boldsymbol{x}_{\boldsymbol{M}_i}, x_i) \log \frac{\mathrm{P}(c \mid \boldsymbol{x}_{\boldsymbol{M}_i}, x_i)}{\mathrm{P}(c \mid \boldsymbol{x}_{\boldsymbol{M}_i})} && (7.4)\\
&= \sum_{\boldsymbol{x}_{M_i}, x_i} \sum_c \mathrm{P}(\boldsymbol{x}_{\boldsymbol{M}_i}, x_i) \cdot \mathrm{P}(c \mid \boldsymbol{x}_{\boldsymbol{M}_i}, x_i) \log \frac{\mathrm{P}(c \mid \boldsymbol{x}_{\boldsymbol{M}_i}, x_i)}{\mathrm{P}(c \mid \boldsymbol{x}_{\boldsymbol{M}_i})} && (7.5)\\
&= \sum_{c, \boldsymbol{x}_{M_i}, x_i} \mathrm{P}(c, \boldsymbol{x}_{\boldsymbol{M}_i}, x_i) \log \frac{\mathrm{P}(c \mid \boldsymbol{x}_{\boldsymbol{M}_i}, x_i)}{\mathrm{P}(c \mid \boldsymbol{x}_{\boldsymbol{M}_i})} && (7.6)
\end{aligned}$$

$$= \sum_{c, \boldsymbol{x}_{M_i}, x_i} \mathrm{P}(c, \boldsymbol{x}_{\boldsymbol{M}_i}, x_i) \log \frac{\mathrm{P}(c \mid \boldsymbol{x}_{\boldsymbol{M}_i}, x_i)\mathrm{P}(x_i \mid \boldsymbol{x}_{\boldsymbol{M}_i})}{\mathrm{P}(c \mid \boldsymbol{x}_{\boldsymbol{M}_i})\mathrm{P}(x_i \mid \boldsymbol{x}_{\boldsymbol{M}_i})} \quad (7.7)$$

$$= \sum_{c, \boldsymbol{x}_{M_i}, x_i} \mathrm{P}(c, \boldsymbol{x}_{\boldsymbol{M}_i}, x_i) \log \frac{\mathrm{P}(c, x_i \mid \boldsymbol{x}_{\boldsymbol{M}_i})}{\mathrm{P}(c \mid \boldsymbol{x}_{\boldsymbol{M}_i})\mathrm{P}(x_i \mid \boldsymbol{x}_{\boldsymbol{M}_i})} \quad (7.8)$$

$$= MI(C; X_i \mid \boldsymbol{M}_i) \quad (7.9)$$

∎

Given Thm. 9, it is trivial to show that if the Markov blanket $\boldsymbol{M}_i$ for feature X_i is $\emptyset$, then $\Delta_{\boldsymbol{Y}}(X_i \mid \boldsymbol{M}_i) = \Delta_{\boldsymbol{Y}}(X_i) = MI(C; X_i)$. Thus, if we harness no conditioning information (i.e., assume that each feature's influence on the class variable is not affected by any other feature) our feature selection measure reduces to the mutual information between the class and each variable. While this simpler measure has been used for feature selection in previous work [153, 175], our derivation places it on a firmer theoretical foundation with respect to the classification task. Moreover, we show how conditioning information should be incorporated into this measure and the underlying assumption of independence made when such information is not used.

Returning to Eq. 7.2, we can make use of this approximation to define the following feature selection algorithm.

Procedure SupervisedFeatureSelection

1. $\forall i, j \; i \neq j$ Compute $\gamma_{ij} \leftarrow \Delta_{\boldsymbol{Y}}(X_i \mid X_j)$
2. $\boldsymbol{R} \leftarrow \boldsymbol{X}$
3. while $(|\boldsymbol{R}| > n_r)$ do
   - 3.1. For each feature $X_i \in \boldsymbol{R}$ do
     - 3.1.1. $\boldsymbol{M}_i \leftarrow$ the set of K features $\{X_{j_1}, \ldots, X_{j_K}\}$ in $\boldsymbol{R} - \{X_i\}$ for which γ_{ij} is smallest.
   - 3.2. $\forall X_i$ Compute $\Delta_{\boldsymbol{R}}(X_i \mid \boldsymbol{M}_i)$.
   - 3.3. $X_{elim} \leftarrow X_i$ for which $\Delta_{\boldsymbol{R}}(X_i \mid \boldsymbol{M}_i)$ is minimal
   - 3.4. $\boldsymbol{R} \leftarrow \boldsymbol{R} - \{X_{elim}\}$.

This algorithm is simple and fairly easy to implement. However, it is clearly suboptimal in many ways, particularly due to the very naive approximations that it uses. We now discuss the consequences of this and some ways in which the algorithm can be improved. First, the current algorithm eliminates a pre-specified number of features (determined by the user set parameter n_r), and constructs sets $\boldsymbol{M}_i$ of a fixed pre-specified size K. It is easy to have the algorithm stop automatically when the expected relative entropy estimate for dropping any remaining feature gets too large. It is also fairly straightforward to extend the algorithm to pick a different size $\boldsymbol{M}_i$ based on the number of features which were highly correlated with X_i. There is, however, an important tradeoff that must be kept in mind. Theoretically, the larger the conditioning set, the likelier it is to subsume all of the information in the feature, thereby forming a Markov blanket. On the other hand, larger conditioning sets fragment our training set into small chunks (corresponding to the different assignment of values to the features in $\boldsymbol{M}_i$), reducing the accuracy of our probability, and hence relative entropy, estimates. Therefore, it is crucial, when doing this modification, to have a penalty term associated with adding additional features to $\boldsymbol{M}_i$. While we do not pursue the issue of using different size Markov blankets for each feature, we do experiment with automatically determining the number of features to eliminate by comparing the increase in relative entropy as features are eliminated with the classification performance of a learning algorithm. We further note that simple cross-validation over a range of feature set sizes can also help determine a good number of features to eliminate with this algorithm.

More importantly, our techniques for choosing the candidate Markov blankets $\boldsymbol{M}_i$ and for evaluating how close each one is to fulfilling the Markov blanket assumption require closer scrutiny. In particular, the expected relative entropy does not really test for the Markov blanket property. The expected relative entropy will also have value 0 if X_i is conditionally independent of C given $\boldsymbol{M}_i$, but we have already pointed out that conditional independence is a weaker property than the Markov blanket assumption. In fact, using conditional independence as a selection criterion can lead to counterintuitive behavior. For example, as we can see in our results, increasing the size K of the conditioning set can actually cause the results to degrade. While some of this degradation is due to fragmentation of the training set (see below), some of it is caused by the fact that conditional independence is not a monotonic property.

That is, it is possible for a certain feature to be conditionally independent of C given some conditioning set $\boldsymbol{M}$, but strongly correlated with C given a strict superset of $\boldsymbol{M}$. In a way, this fact is not surprising. It is well-known that additional information can cause correlations that were not present before to appear [125].

To illustrate this phenomenon in our context, consider a hypothetical text classification problem, where (as described in Chapter 2) the data instances are documents, the features are the presence or absence of words, and the classes are document topics. The word *mining* is not significantly correlated with the topic *machine-learning*. Therefore, if we were to run our algorithm with $K = 0$, we would probably eliminate *mining* fairly early. However, this word is strongly correlated with the word *data*. Moreover, if we condition on the presence of the word *data*, there is a strong correlation between the word *mining* and the topic *machine-learning*. Thus, by putting the word *data* into our conditioning set $\boldsymbol{M}$, we have caused a seemingly irrelevant word to become relevant. The converse can also occur, so that we can get the estimated "relevance" of a feature fluctuating multiple times as we change K. We believe that the performance of our algorithm will be significantly improved by the use of more refined techniques (e.g., Bayesian methods) to choose a candidate (or even several candidates) Markov blanket, and by the use of a more precise formula for evaluating how close the different candidates are to fulfilling the requirement.

As far as computational expense, our algorithm shows promise for scaling to larger domains. The time complexity of our algorithm is quite low. Theoretically, it requires $O(n^2(m + \log n))$ operations for computing the pairwise relative entropy matrix and sorting it, where n is the initial number of features and m is the number of instances. The subsequent feature selection process requires $O(rnkmcv^k)$ time, where r is the number of features to eliminate, v is the maximum number of distinct values that a feature may take, k is the small, fixed number of conditioning features and c is the number of classes. This second complexity result is due to the fact that to eliminate a single feature, we must iterate through all the remaining features (at most n) and for each one select the k features which are most correlated with it. To then compute $\Delta_{\boldsymbol{R}}(X_i \mid \boldsymbol{M}_i)$ for each feature, requires $O(mcv^k)$ time to iterate through the m data instances and compute the cv^k entries in the conditional probability tables that we must sum over. Using caching schemes, it is possible to reduce the second term by close to a factor of n, due to the fact that an eliminated feature is likely to be in the $\boldsymbol{M}_i$ of only a few of the remaining features. Thus, we need only recompute a new $\boldsymbol{M}_i$ and its expected relative entropy for this small number of features.

7.4 Initial Results on Non-Text Domains

In order to empirically test our theoretical model for feature selection as implemented by our approximate algorithm, we ran a number of experiments on both artificial data as well as real-world data from both textual and non-textual domains. We report first on the results using non-textual data since we have a good a priori understanding of how feature selection should affect this data and we can verify that our method is working well. These datasets include: the Corral data (which was artificially constructed by John, Kohavi, and Pfleger [84] specifically for research in feature selection), as well as the LED24, Vote, and DNA datasets from the UCI repository [119]. These datasets are detailed in Table 7.1.

Dataset	Number of Classes	Number of Features	Training Set Size	Testing Set Size
Corral	2	6	32	128
LED24	10	24	3200	5-fold CV
Vote	2	48*	435	5-fold CV
DNA	3	180*	3186	5-fold CV

Table 7.1: Datasets from the UCI repository used in feature selection experiments. *Reflects Boolean encoding of feature values.

We first analyze the artificial domains. The Corral dataset has been noted by previous researchers [84] as particularly difficult for filter methods since, of the 6 features in this domain, the target concept is a Boolean function of only four of the features: $(A \wedge B) \vee (C \wedge D)$. The fifth feature is entirely irrelevant and the sixth feature is "correlated" with the target concept in that it matches the class label 75% of the time. Thus, many filter approaches which use forward selection are likely to always select the correlated feature. This poses a problem for certain induction methods. The C4.5 decision tree algorithm, for example, is likely to initially split on the correlated feature, thus fragmenting the data enough that the true target concept cannot be recovered in the subtrees. One should note, however, that, due to the disjunctive nature of the target function, the Naive Bayesian classifier is actually better off with the correlated feature than without it. This seems to be more a shortcoming of the simplicity of this induction method than a flaw with feature selection methods that eliminate the correlated feature.

We verified this analysis experimentally, finding that forward selection (even with conditioning) always selects the correlated feature (thereby taking a "large" step in a suboptimal direction, as in Figure 7.1). Running backward elimination with conditioning, on the other hand, avoids this problem: we eliminate the correlated feature, since it has no effect on the class distribution for the function given the features that determine the target concept (or some large subset thereof). When we conditioned on 2 or more features and set the algorithm to drop 2 features, it consistently eliminated both the correlated and irrelevant features.

In the LED24 domain, we find a situation (albeit artificial) where conditioning on correlated features actually makes it more difficult to determine an appropriate subset of features. This domain contains 7 relevant and 17 irrelevant features. Moreover, the class label in the LED24 domain entirely determines the value of each relevant feature (modulo a noise term), whereas the irrelevant features are random. Thus, there is no dependence between features given the class label. As a result, we would expect that conditioning on correlated features would only confuse our algorithm by forcing it to unnecessarily estimate a larger number of probability values with the same amount of data, thus leading to poorer estimates. Again, this conjecture was verified experimentally, as our method in fact only consistently selected the 7 relevant features when we conditioned on no variables.

To initially test how our method of feature subset selection affected classification, we employed both a Naive Bayesian classifier [46] and C4.5 [130] as induction algorithms. These algorithms were applied both to the original datasets and to the datasets filtered through our feature selection algorithm (using both forward selection and backward elimination). Accuracy results (and standard deviations, when cross-validation was used) for the UCI data are given in Tables 7.2 and 7.3.

Dataset	# Features Orig./Final	K	Naive Bayes Accuracy Original	Forward	Backward
Corral	6 / 4	0	90.6	84.4	84.4
		1		81.3	84.4
		2		81.3	87.5
		3		81.3	87.5
		4		81.3	87.5
LED-24	24 / 14	0	72.1 ± 2.1	72.1 ± 1.0	72.1 ± 1.0
		1		71.9 ± 0.9	72.1 ± 0.7
		2		72.2 ± 1.4	72.4 ± 1.4
LED-24	24 / 7	0	72.1 ± 2.1	72.8 ± 1.5	72.8 ± 1.5
		1		72.2 ± 1.8	72.2 ± 1.8
		2		72.1 ± 1.5	72.1 ± 0.8
Vote	48 / 28	0	90.1 ± 1.8	90.1 ± 1.8	90.1 ± 1.8
		1		90.1 ± 2.7	90.1 ± 2.7
		2		90.3 ± 3.4	90.3 ± 3.4
Vote	48 / 8	0	90.1 ± 1.8	92.0 ± 2.7	92.0 ± 2.7
		1		93.6 ± 1.8	92.7 ± 2.5
		2		95.2 ± 2.6	93.1 ± 5.4
DNA	180 / 80	0	94.0 ± 0.6	95.0 ± 0.5	95.0 ± 0.5
		1		95.4 ± 0.7	95.5 ± 0.9
		2		94.9 ± 0.9	94.8 ± 0.6
DNA	180 / 30	0	94.0 ± 0.6	94.1 ± 1.0	94.1 ± 1.0
		1		94.2 ± 1.1	94.3 ± 1.1
		2		92.3 ± 1.8	93.8 ± 1.0

Table 7.2: Accuracy percentages for Naive-Bayes using feature selection.

As seen in the accuracy results for Corral (using C4.5), Vote (using Naive Bayes with aggressive feature elimination), and DNA, selection of the appropriate feature set can have a large impact on classification accuracy. In the DNA domain we see some of the most dramatic results: accuracy improvements after eliminating 100, or even 150, of the 180 features with our method! A two-tailed paired T-test over

Dataset	# Features Orig./Final	K	C4.5 Accuracy Original	Forward	Backward
Corral	6 / 4	0	81.2	81.2	75.0
		1		75.0	87.5
		2		75.0	100.0
		3		81.2	100.0
		4		81.2	100.0
LED-24	24 / 14	0	71.1 ± 1.2	71.3 ± 1.2	71.3 ± 1.3
		1		71.0 ± 1.0	70.9 ± 1.2
		2		71.9 ± 1.0	71.3 ± 0.9
LED-24	24 / 7	0	71.1 ± 1.2	72.1 ± 0.9	72.1 ± 0.9
		1		71.3 ± 1.9	71.3 ± 1.9
		2		71.6 ± 1.3	71.5 ± 1.1
Vote	48 / 28	0	95.2 ± 1.5	95.7 ± 1.5	95.7 ± 1.5
		1		95.0 ± 2.8	95.2 ± 2.8
		2		94.5 ± 1.3	94.7 ± 1.3
Vote	48 / 8	0	95.2 ± 1.5	95.4 ± 2.9	95.4 ± 2.9
		1		95.9 ± 1.5	95.7 ± 1.5
		2		95.0 ± 2.5	96.0 ± 2.2
DNA	180 / 80	0	92.3 ± 0.7	93.6 ± 0.7	93.5 ± 0.7
		1		93.4 ± 0.8	93.3 ± 0.6
		2		93.6 ± 1.4	93.5 ± 1.1
DNA	180 / 30	0	92.3 ± 0.7	93.6 ± 0.2	93.6 ± 0.3
		1		93.6 ± 0.8	93.4 ± 0.7
		2		91.7 ± 1.5	93.3 ± 1.0

Table 7.3: Accuracy percentages for C4.5 using feature selection.

the cross-validation folds reveals statistically significant improvements ($P < 0.10$) in accuracy for the Vote domain using Naive Bayes with aggressive feature selection and in the DNA domain for backward elimination used in conjunction with C4.5. Moreover, feature selection never significantly degraded accuracy in any of the real-world datasets from UCI tested. More importantly, however, is the fact that, in many domains, our feature selection algorithm can make dramatic reductions in the feature space and consequently improve running-time performance as well.

Empirically, the low running time of our algorithm allows us to deal with very large domains in a reasonable amount of time. By way of comparison, Kohavi [90] obtains similar accuracy results on the DNA dataset for Naive-Bayes and C4.5 using the wrapper approach, but notes that doing so takes 15 hours on a Sun SPARC 10. In our experiments, an inefficient implementation of our algorithm (one that did not utilize clever data structures to reduce the running time) reduced the DNA dataset by 100 features using between 10 and 20 minutes on the same machine [1] (depending on the number of conditioning variables). This is a time savings of nearly two orders of magnitude! Moreover, since our approach is a filter method, we do not need to re-run the algorithm for every induction algorithm we choose to run on a reduced-feature dataset.

7.5 Results on Text Domains

Returning to our original motivation for this work, we now evaluate our feature selection algorithm on the high-dimensional datasets common in text domains. The dimensionality of such datasets present an exceptional challenge for many feature selection algorithms. In particular, feature selection using a wrapper method is simply intractable due to the prohibitive cost of running an induction algorithm thousands of times on this very high-dimensional data. Hence, an efficient filter method, akin to the one method described here, is the only suitable approach.

We begin by considering two new subsets of the Reuters document collection

[1] Actually, the machine we ran our initial experiments on is *exactly* the same machine (named "Starry") that Kohavi used in his study.

Data set	Number of Docs	Number of Words	Categories	Testing Method
Reuters1	337	1675	Coffee, Iron-Steel, Livestock	5-fold CV
Reuters2	379	1646	Reserves, Gold, GNP	5-fold CV

Table 7.4: Subsets of the Reuters document collection used for initial feature selection experiments.

[132] and then turn our attention back to the five subsets of the Reuters collection (D1 - D5) initially presented in Chapter 5. The two new datasets, named Reuters1 and Reuters2, each contain three topics (classes) and are described in more detail in Table 7.4.

The first subset, Reuters1, is comprised of topics that are not likely to have many meaningful overlapping words. Reuters2, on the other hand, contains topics which are likely to have many similar words used in different contexts across topics. In accordance with Zipf's Law, all words which occurred less than 3 times in each dataset were eliminated simply as a means for removing extremely rare words such as unique names. We ran our feature selection algorithm on these two Reuters datasets in order to reduce the feature space by 1000 features — down to nearly 1/3 its original size — and get an initial feel for how well this algorithm would work in text domains. The results of these experiments are shown in Tables 7.5 and 7.6, which also includes results for forward selection. The best results for each dataset is shown in bold face in each table.

In the Reuters1 domain, where we expect more distinct terms between topics (and hence less feature interaction) we see that both feature selection methods have a tendency to work comparably well without conditioning information, producing good accuracy results. When conditioning is introduced, however, the results using the backward elimination method clearly dominate those obtained using forward selection. Employing forward selection is simply inadequate for finding good features with conditioning information.

In the second Reuters domain, we see that employing backward elimination allows

Dataset	# Features Orig./Final	K	Naive Bayes Accuracy Original	Forward	Backward
Reuters1	1675 / 675	0	90.5 ± 5.2	90.7 ± 5.1	90.7 ± 5.1
		2		91.9 ± 3.1	**96.1 ± 3.3**
Reuters2	1646 / 646	0	89.9 ± 2.2	92.3 ± 1.7	92.3 ± 1.7
		2		90.6 ± 2.5	**94.1 ± 2.8**

Table 7.5: Accuracy percentages for two Reuters text datasets using Naive Bayes and feature selection.

Dataset	# Features Orig./Final	K	C4.5 Accuracy Original	Forward	Backward
Reuters1	1675 / 675	0	95.2 ± 2.5	95.5 ± 2.4	94.9 ± 2.7
		2		95.8 ± 2.5	**96.4 ± 2.0**
Reuters2	1646 / 646	0	91.5 ± 1.2	93.0 ± 0.6	93.0 ± 0.6
		2		91.5 ± 2.9	**94.9 ± 2.0**

Table 7.6: Accuracy percentages for Reuters text datasets using C4.5 and feature selection.

the algorithm to make effective use of conditioning information to increase the accuracies of both induction methods in a drastically reduced feature space. The results for Reuters2 using our backward elimination method with $K = 2$ are statistically significant ($P < 0.10$) improvements over the accuracy on the original dataset. Indeed, the combination of backward selection and using conditioning information leads to the best accuracy results for both datasets, regardless of whether C4.5 or Naive Bayes is used as the subsequent learning method.

As for resource consumption, eliminating 1000 features from the Reuters datasets while making use of conditioning information took about 2.5 hours on a Sun SPARC 10. If we were not to use such conditioning information (i.e., set $K = 0$), the time required for feature selection could be reduced to under a minute, making the algorithm quite practical for use in an interactive system such as SONIA. By way of comparison, a rough estimate of the time required by a wrapper approach, such as that of Caruana and Freitag [20] or John, Kohavi, and Pfleger [84], to eliminate this many features is on the order of thousands of hours, assuming the method does not get caught in a local minimum first and prematurely stops eliminating attributes as a result.

With these initial results on text data giving us a better understanding of how well our algorithm performs in such domains, we now consider its efficacy on the five text datasets D1 through D5. Since our previous results showed that backward elimination generally outperforms forward selection, we focus only on the former method in our follow-up experiments. Furthermore, since we wish to stay in the realm of probabilistic classification methods, we only consider the Naive Bayesian classifier at this point and no longer report results using C4.5.

We ran our feature selection algorithm with various levels of conditioning ($K = 0, 1$, or 2), producing feature sets of size 20, 50, 100, 200, and 400 for each dataset. We used 10-fold cross-validation to obtain classification accuracies for each such run. The results of these experiments are plotted in Figures 7.2 through 7.6, corresponding to datasets D1 through D5, respectively. In these figures, we also report the average accuracy over the five reduced feature set sizes, as well as the accuracy obtained when the entire feature set is used (which shows up as a straight line in each plot).

It is worth noting that in virtually all of the text datasets we are able to reduce the sizes of the feature sets to less than 5% of their original sizes without a significant degradation in accuracy. In fact, in several cases we actually obtain significantly better results. Starting with D1, for example, we can reduce the feature set to 50 features (less than 5% of the original feature space) while producing virtually no reduction in accuracy. In fact, we see a slight increase in accuracy on this dataset until we reach 200 features. In D2, D3, and D4, we see an improvement in classification accuracy across the entire range of feature sizes tested, with the improvements often becoming more pronounced for the smallest feature set sizes. In fact, for D3 with 20 features, and D4 with 50 features (with $K = 0$ or 1 in both cases) the accuracy improvements are statistically significant using a paired t-test ($P < 0.10$). Only in D5, which has the largest initial feature set (nearly 2000 features) do we see feature selection consistently producing inferior results to using the full feature set. Still, it is worth noting that only at the smallest numbers of features (20 and 50 features, which

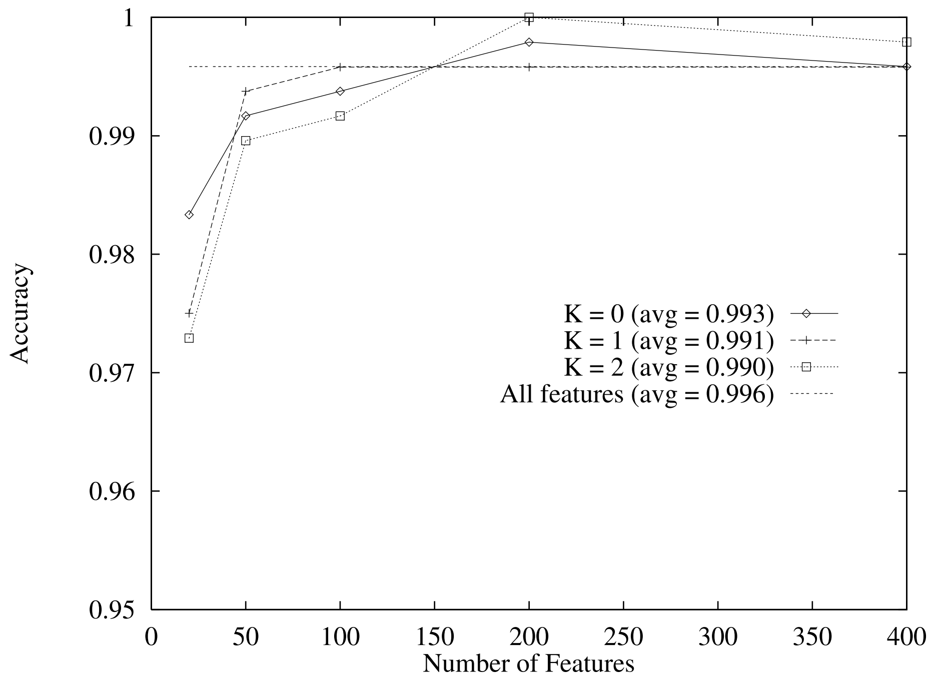

Figure 7.2: Results on Dataset 1 using Naive Bayes with feature selection.

comprise roughly 1% and 2.5% of the original feature set, respectively) are the results actually significantly worse. On balance, when averaging over all five datasets, we find that feature selection generally provides modest improvements in classification accuracy.

We also tried to measure the extent to which the expected relative entropy score can be used to automatically determine an appropriate number of features for each dataset. To this end, we considered the gain in information over using the simple class prior distribution $P(C)$ as measured by the expected relative entropy for different size feature sets. The results of these measurements for the datasets D1 though D5 are given in Figure 7.7. Most striking in this figure is the difference in gain between D5 and the other datasets. This discrepancy would seem to indicate that on average each feature in D5 contains much less information about a document's class than the features in the other datasets (perhaps because D5 contains far more documents and initial features than the other text datasets). Hence, more features would be needed for accurate classification in D5. This conclusion is clearly borne out in the accuracy

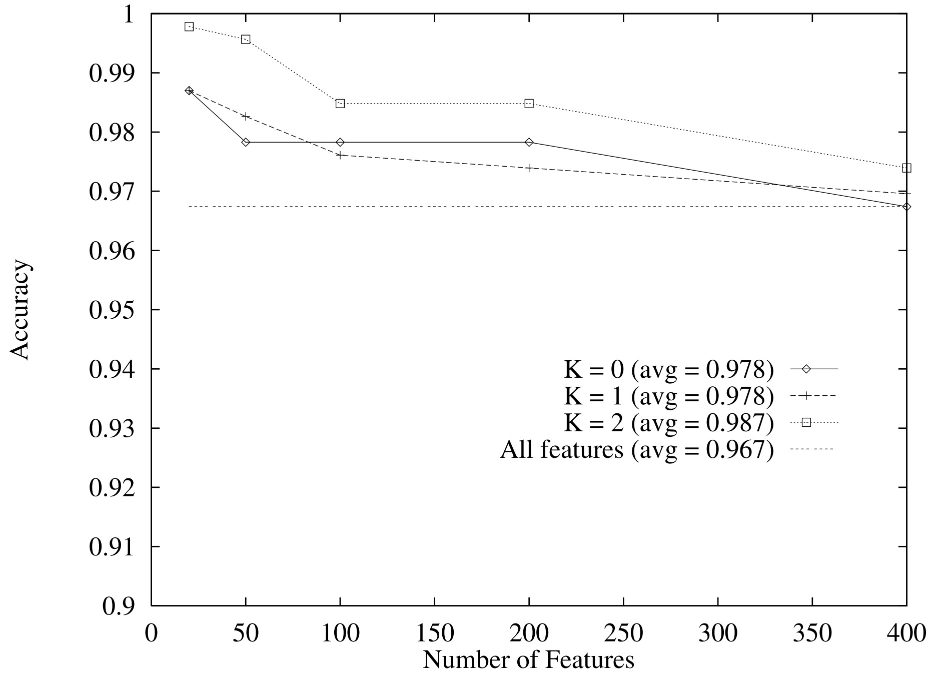

Figure 7.3: Results on Dataset 2 using Naive Bayes with feature selection.

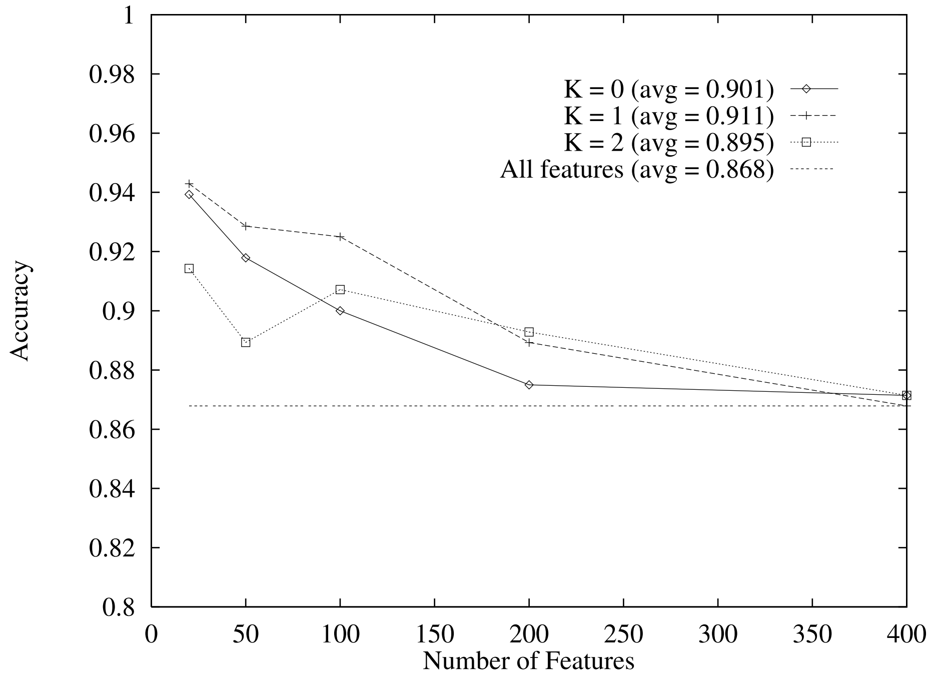

Figure 7.4: Results on Dataset 3 using Naive Bayes with feature selection.

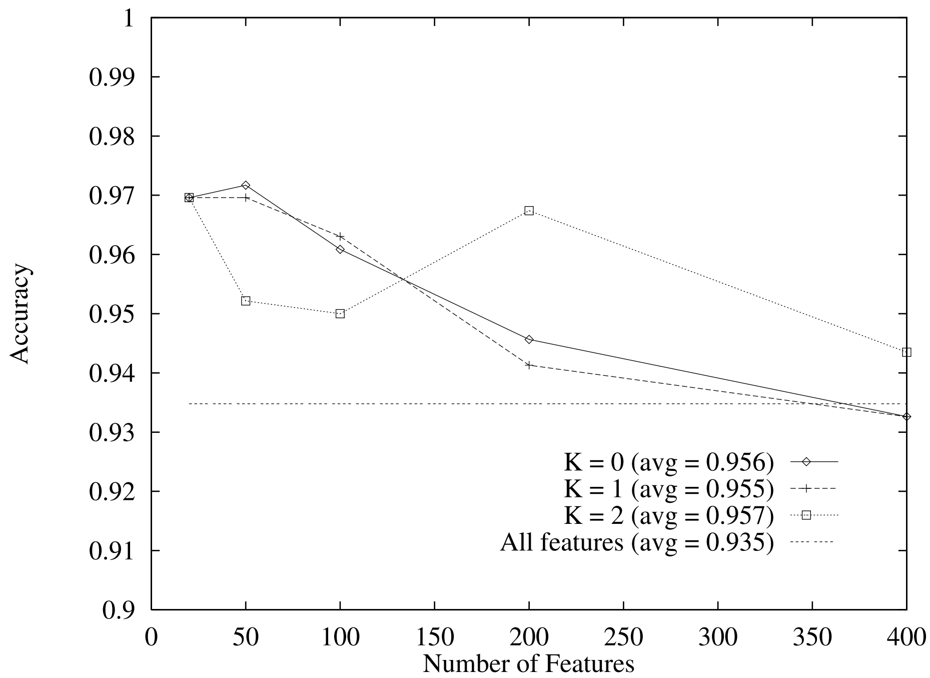

Figure 7.5: Results on Dataset 4 using Naive Bayes with feature selection.

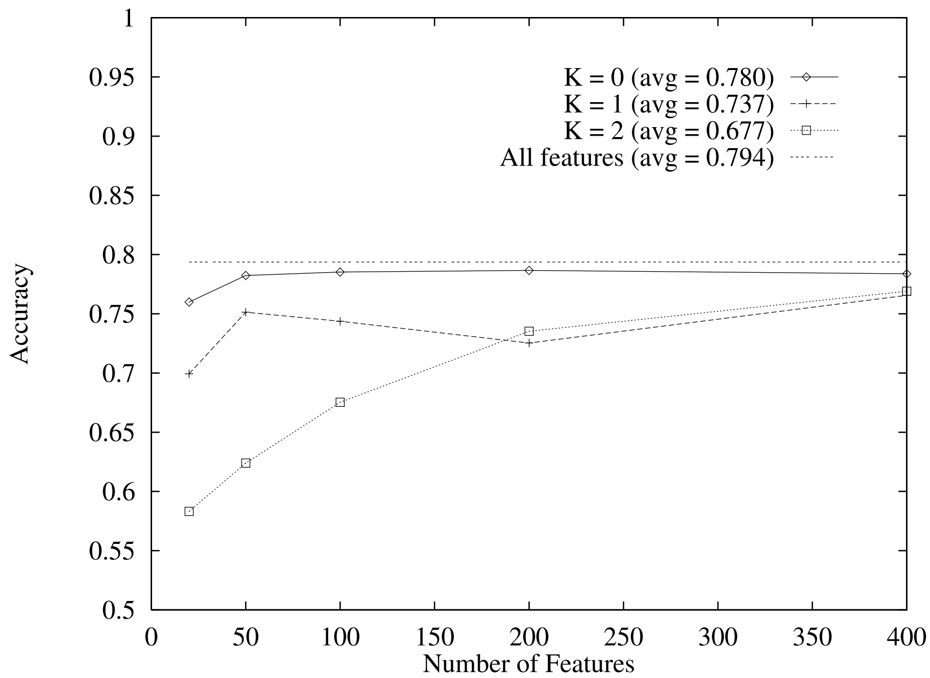

Figure 7.6: Results on Dataset 5 using Naive Bayes with feature selection.

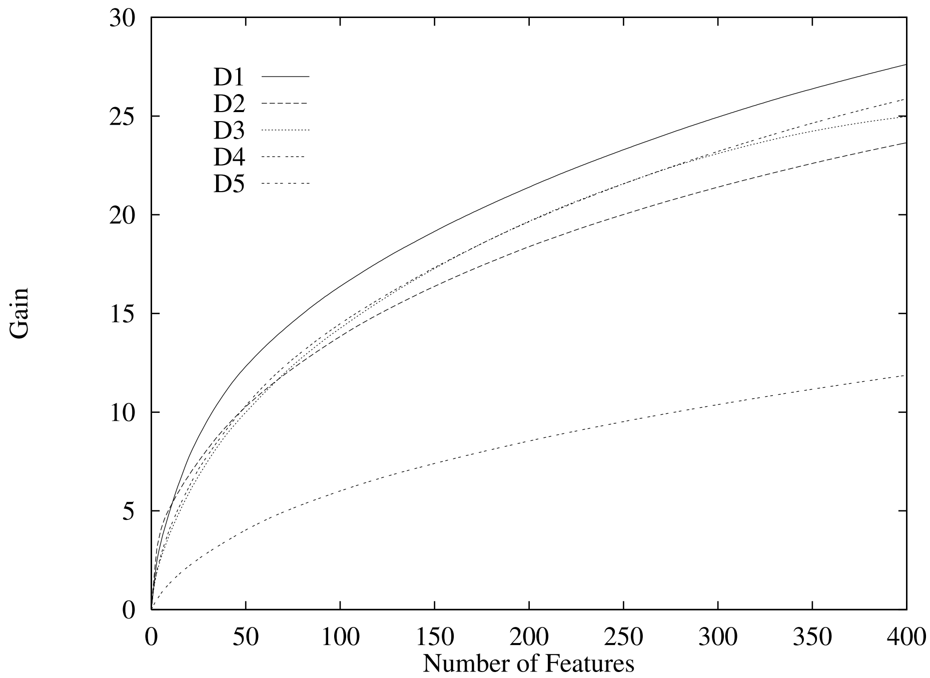

Figure 7.7: Gain in bits over P(C) with different size feature sets.

results presented earlier. In fact, to realize the same level of gain reached in D5 with 400 features, we would only need approximately 50 features in D1, and approximately 70 features in D2, D3, and D4. While these exact feature set sizes are not quite optimal, as reflected in the accuracy figures presented earlier, they nevertheless are in a very good ballpark. Hence, we believe that the expected relative entropy measure can be used as a means for finding the right general range of feature set sizes that produce good classification accuracies. Further refinements to the feature set size can then be realized by using cross-validated classification accuracy over this small ballpark range.

7.6 Conclusions

We have presented a theoretically justified model for optimal feature selection based on using relative entropy to minimize the amount of predictive information lost during feature elimination. Within this theoretical framework, we prove several desirable properties of using such a method for feature selection. Moreover, we present an algorithm that roughly approximates our theoretical model and provide extensive empirical testing. We show that this algorithm is effective at drastically reducing the feature space in many learning tasks while also helping to improve accuracy in many cases.

Our method attempts to eliminate features in a way that keeps the conditional probability of the class given the features as close to the original distribution as possible. This approach is not the same as attempting to maintain the same *classification* for each instance. While this too is a desirable goal, it is necessarily specific to a particular induction algorithm. Rather, we focus on an algorithm-independent paradigm for feature subset selection, viewing an induction algorithm as a biased method for approximating the probability distribution of class labels given features and transforming this distribution into a classification. We stay free of the bias of a particular induction algorithm by simply maintaining as much as possible the underlying conditional distribution of class labels that the induction algorithm attempts to approximate.

Due in large part to its induction-bias-free nature, our approach sometimes provides only modest gains in accuracy. However, improving classification accuracy is not our primary goal at this point. Rather, we utilize feature selection as a means for allowing more expressive (and often more computationally expensive) induction algorithms to be applied to high-dimensional text domains. Currently, when faced with such a domain, we are essentially forced to use simple, less computationally intensive induction algorithms such as Naive Bayes. The use of our feature selection algorithm as a pre-processing step will enable the use of more powerful induction algorithms, such as Bayesian classifiers which make use of feature dependence information. We explore this issue more fully in next two chapters. In this way, we hope to make such induction methods much more applicable to classification problems with many features. Furthermore, although we argue that wrapper methods are, a priori, too computationally expensive for text datasets, we can use them on the feature-reduced datasets resulting from our algorithm. This will allow us to produce a classifier which is optimized for accuracy with respect to a specific induction algorithm, by searching in the *filtered* feature space. We hope that these techniques, taken together, will allow us to further tackle induction problems in very large feature spaces.

Chapter 8

Limited Dependence Bayesian Classifiers

8.1 Introduction

In order to more accurately classify text documents, it is necessary to use an induction algorithm whose bias corresponds with our intuitions about textual domains. Addressing this issue in the context of probabilistic classifiers, we now turn our attention to methods for learning Bayesian classifiers which better model the feature dependencies that exist in text.

As mentioned in Chapter 3, work in Bayesian methods for classification has grown enormously of late [31, 72, 17, 52]. Nevertheless, the notion of Bayesian classification at least dates back to the Naive Bayesian classifier [64], which has existed since the early days of pattern recognition research. This classifier has had a longer history as a simple, yet powerful, classification technique and continues to be used widely today.

Although learning of general Bayesian networks as well as the Naive Bayesian classifier have both shown success in different domains, each has its shortcomings. Learning in the domain of unrestricted Bayesian networks is often very time consuming and quickly becomes intractable as the number of features in a domain grows. Indeed, it has been proven that learning optimal unrestricted network structures is NP-hard [25]. Moreover, inference in such unrestricted models has also been shown to be NP-hard [30].

Alternatively, the Naive Bayesian classifier, while very efficient for inference, makes very strong independence assumptions that are often violated in practice, especially in text domains, and can lead to poor predictive generalization. In this chapter, we seek to identify the limitations of each of these methods, and show how they represent two extremes along a spectrum of Bayesian classification algorithms. We then present an algorithm for learning Bayesian network classifiers that relaxes the restrictive Naive Bayes assumption and allows for efficient learning of models with limited feature dependency structure. The ability to learn feature dependency models efficiently makes this method particularly useful for text classification problems. Still, the algorithm is general enough to use on any domain.

In Section 8.2 we formally describe the probabilistic models alluded to above. Section 8.3 presents the KDB algorithm that allows us to move along a spectrum between these two extremes, allowing us to appropriately model data based on our beliefs about the existence of dependencies in a domain. Empirical results using this algorithm on both standard UCI domains as well as text classification problems are presented in Sections 8.4 and 8.5, respectively. Finally, Section 8.6 presents our conclusions and more details of related work in probabilistic classification.

8.2 Probabilistic Classification Models

To lay the foundation for understanding the spectrum of classification models, we begin by first examining the end points of this spectrum and then naturally generalize.

8.2.1 Unrestricted Bayesian Classifiers

Recall that in probabilistic classification, we would ideally like to determine the probability distribution $\mathrm{P}(C \mid \boldsymbol{X})$ where C is the class variable and $\boldsymbol{X}$ is the n-dimensional vector of variables $(X_1, X_2, \ldots, X_n)$ whose instantiation $\boldsymbol{x}$ represents an observed instance. If we had this true distribution available to us, we could achieve the theoretically optimal classification (with respect to a uniform loss function over classes)

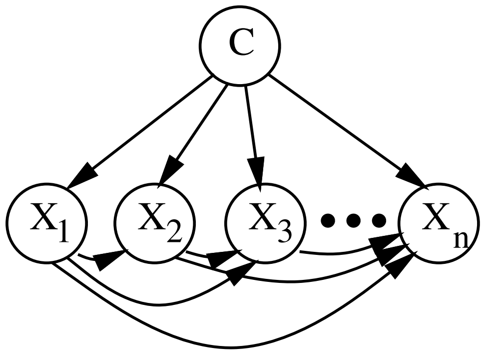

Figure 8.1: Bayesian network representing $\mathrm{P}(\boldsymbol{X} \mid C)$, allowing for complete dependency between domain features.

by simply classifying each instance $\boldsymbol{x}$ into the class c_k for which $\mathrm{P}(C = c_k \mid \boldsymbol{X} = \boldsymbol{x})$ is maximized. This is known as *Bayes Optimal* classification.

By applying Bayes Law to $\mathrm{P}(C = c_k \mid \boldsymbol{X} = \boldsymbol{x})$, we obtain

$$\frac{\mathrm{P}(\boldsymbol{X} = \boldsymbol{x} \mid C = c_k) \cdot \mathrm{P}(C = c_k)}{\mathrm{P}(\boldsymbol{X} = \boldsymbol{x})} \ . \tag{8.1}$$

Since the denominator in Eq. 8.1 does not depend on the choice of class c_k, it can be safely ignored when selecting the class which maximizes this expression. Thus, Bayes classification requires us to determine only the two probabilities $\mathrm{P}(\boldsymbol{X} = \boldsymbol{x} \mid C = c_k)$ and $\mathrm{P}(C = c_k)$. While the distribution $\mathrm{P}(C = c_k)$ may be determined in a straightforward way, the distribution $\mathrm{P}(\boldsymbol{X} = \boldsymbol{x} \mid C = c_k)$ is more problematic to define.

In the completely unrestricted case, we may model $\mathrm{P}(\boldsymbol{X} = \boldsymbol{x} \mid C = c_k)$ by allowing for arbitrarily many dependencies between features. In the most extreme form of such an unrestricted network, every feature is dependent on every other feature. The Bayesian network shown in Figure 8.1 corresponds to this extreme case. In this figure, we can see that the true complexity in such *full-dependence* model comes from the large number of feature dependence arcs which are present in the model. Learning such a model, especially in domains with many features, can be prohibitive.

There has been much recent research in learning in the context of generally unrestricted Bayesian networks [72]. These learning methods attempt to model the full joint probability distribution for a set of features, but rely on identifying feature independencies in order to keep computation tractable. Nevertheless, as mentioned earlier, the process of learning the appropriate graphical structure representing the probabilistic dependencies inherent in a dataset remains very expensive computationally. Moreover, most work in this area has focused on primarily on density estimation and it is only very recently that such models have been tailored especially for use with supervised classification problems [53, 152].

Nevertheless, since unrestricted Bayesian networks (as shown in Figure 8.1) allow for the modeling of arbitrarily complex dependencies between features, we can think of these extreme models as lying at the most general end of a *feature dependence spectrum*. Such Bayesian classifiers have great representational power at the cost of computational expense.

8.2.2 Naive Bayes

The Naive Bayesian classifier (also known as *Idiot* Bayes) represents the most restrictive extreme in the spectrum of probabilistic classification techniques. It employs the very restrictive assumption that each feature X_i is conditionally independent of every other feature given the class label, when predicting the class c_k that maximizes $\mathrm{P}(C = c_k \mid \boldsymbol{X} = \boldsymbol{x})$. Recall that this assumption is more formally written as

$$\mathrm{P}(\boldsymbol{X} = \boldsymbol{x} \mid C = c_k) = \prod_i \mathrm{P}(X_i = x_i \mid C = c_k) \ . \tag{8.2}$$

The network structure of the Naive Bayes model is shown in Figure 8.2. In contrast to Figure 8.1, we see that the Naive Bayes model allows for no arcs between feature nodes (although the arcs from C to the features X_i appear in both models). We can think of Naive Bayes as being at the most restrictive end of the feature dependence spectrum, in that it strictly allows no dependencies between features given the class label. While the conditional independence assumption is unrealistic in many domains, especially text, the Naive Bayesian classifier surprisingly often gives good classification

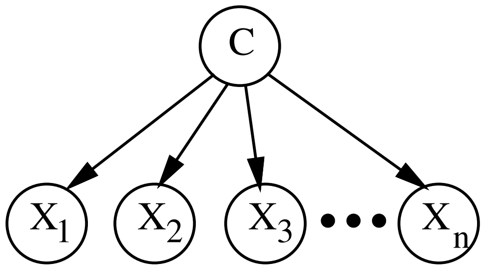

Figure 8.2: Bayesian network representing a Naive Bayesian classifier.

accuracy. In domains in which feature dependencies may play a key role, however, the Naive Bayesian algorithm can fail to perform well.

Naive Bayes has been very popular due to its computational efficiency. Training such a classifier only requires time that is linear in the number of features and data instances. The maximum likelihood values for the necessary probabilities can be easily estimated directly from counts in the data. In many cases, simple priors are also incorporated into these estimates in a straightforward manner (e.g., Laplace estimation [91]).

The simplicity of this classifier has also made it the beneficiary of a number of recent research efforts. Work in this area includes average case analyses of error rates using Naive Bayes [103], as well as methods for handling features with continuous values, such as discretizing the data into suitable intervals [44] or performing some form of univariate kernel density estimation [85].

8.2.3 Limited Dependence Classifiers

We now formalize our notion of the "spectrum of feature dependence" in Bayesian classification by introducing the notion of k-dependence Bayesian classifiers.

Definition 3 *A* k-dependence Bayesian classifier *is a Bayesian network which contains the structure of the Naive Bayesian classifier and allows each feature* X_i *to have a maximum of* k *feature nodes as parents. That is,* $\Pi(X_i) = \{C, \boldsymbol{X}_{d_i}\}$ *where* $\boldsymbol{X}_{d_i}$ *is a set of at most* k *feature nodes, and* $\Pi(C) = \emptyset$*.*

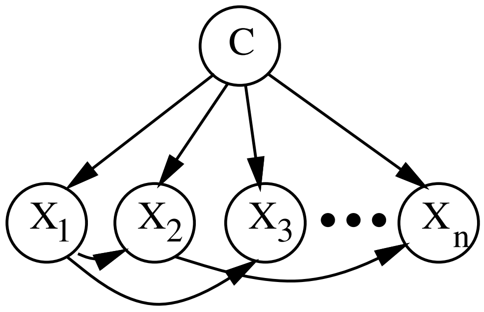

Figure 8.3: Bayesian network representing a 1-Dependent Bayesian classifier.

Observation 10 *The Naive Bayesian classifier is a* 0*-dependence Bayesian classifier.*

Observation 11 *The fully unrestricted Bayesian classifier (i.e., no independencies) is a* $(n-1)$*-dependence Bayesian classifier, where* n *is the number of domain features.*

By varying the level of allowable dependence k in a k-dependence Bayesian classifier, we can define models that smoothly move along the spectrum of feature dependence. An example of a 1-dependence Bayesian classifier is shown in Figure 8.3. Note that such 1-dependence models capture the notion of *tree structured dependency models* described in Chapter 5 (modulo the difference that here we are dealing directly with a classification task, and hence have a class variable C that all features are also conditioned on).

If k is large enough to capture all feature dependencies that exist in a domain, then we could expect a classifier to achieve Bayes optimal accuracy if the "right" dependencies are set in the model and the model parameters could be determined accurately. The question becomes one of determining if the model has allowed for enough dependencies to represent the *Markov blanket* [125] of each feature. The subtleties of this point are discussed in more detail in Chapter 7 as well as the work of Friedman and Goldszmidt [53]. In practice, however, the limited size of a dataset relative to the entire instance space (which grows exponentially with the number of features) makes it extremely difficult to learn the "true" model and thus achieve Bayes optimal classification.

8.3 The KDB Algorithm

In this section, we give an algorithm, named KDB (for k-Dependence-Bayes), which allows us to construct classifiers at arbitrary points (values of k) along the feature dependence spectrum, while also capturing much of the computational efficiency of the Naive Bayes classifier. From the analysis of probabilistic methods in Section 8.2, it becomes apparent that the amount of feature dependence that is allowed in a Bayesian network classifier is the key to the modeling power of the classifier—this is how we define the feature dependence spectrum. Unfortunately, as this degree of dependence grows, the cost of generating such classifiers can increase dramatically. Indeed, the NP-hardness results for learning Bayesian networks cited in Section 8.1 apply to learning *optimal* k-dependence Bayesian classifiers for $k \geq 2$.

Consequently, we present an alternative to the general trend in Bayesian network learning algorithms which do an expensive search through the space of network structures [72] to find a locally optimal networks. Rather, focusing on the issue of tractable computation, we provide an algorithm which heuristically constructs good, but potentially suboptimal, Bayesian network structures efficiently.

Before delving into the details of our algorithm, however, we provide some context for our development choices by briefly discussing some of the previous work in this area that is most directly related to our algorithm. One of the earliest algorithms for actually learning Bayesian network structure is the well-known K2 algorithm of Cooper and Herskovits [31]. While K2 bears some superficial similarities to our method, it was developed as an unsupervised learning method (i.e., when no class information is provided) to model a general joint probability distribution. In our algorithm, we specifically aim to generate a classifier, and thus focuses on supervised learning problems by making use of class dependent measures as heuristics to guide network construction. This difference in focus can have a large impact on the kinds of network structures that are produced by our method as opposed to those produced by K2 (for more details, see the discussion in Section 8.6). Furthermore, K2 requires an explicit ordering of the features as input. To relax this requirement, our algorithm produces its own feature ordering, again making use of classification specific measures to determine the order.

Looking specifically at models that limit the amount of feature dependence, Geiger [60] has defined the notion of a *conditional dependence tree*, which corresponds in our framework to a 1-dependence model that does not include a class variable. Such models can be learned optimally in $O(n^2)$ time, where n is the number of domain features, using a method directly based on the work of Chow and Liu [26].

More recently, Friedman and Goldszmidt [53] have independently developed an algorithm (described briefly in Chapter 3), named TAN (*T*ree *A*ugmented *N*aive-Bayes), which is similar to Geiger's method for inducing conditional trees, but is specifically targeted at the classification task. This algorithm can efficiently (in $O(n^2)$ time) generate optimal 1-dependence Bayesian classifiers. The method has shown impressive results in experimental comparisons with Naive Bayes. However, it provides no straightforward extension to generalize to higher degrees of feature dependence. We would like our method to be of similar computational complexity to TAN, but be able to represent more complex feature dependencies.

Ezawa and Schuermann [49] have developed an algorithm APRI, which predates ours, and also performs a heuristic search to find Bayesian classifiers with higher order feature interactions. While our algorithm makes use of the same measures as APRI for determining the dependencies between features in the Bayesian network, APRI does not explicitly bound the degree of dependence that may be modeled by an induced classifier. As a result, this algorithm provides no guarantee that it will produce a strictly k-dependence classifier. This lack of an explicit complexity control can be especially problematic in text domains, where the feature space is so large that an explicit control on the level of dependence in the network is critical to prevent the network from becoming unwieldy.

With this previous work in mind, the goal of our algorithm is to allow for various levels of feature dependency to be learned efficiently, while also maintaining an explicit control on the network complexity. In this way, we hope to learn flexible, yet explicitly bounded, models that are more accurate than the Naive Bayesian classifier. However, we point out that the main goal of developing our algorithm is not to attempt to show that it is superior (in terms of classification accuracy) to the other methods for learning Bayesian networks just mentioned. Hence, we make no experimental comparisons with these methods. Rather, our main motivation in this regard is simply to show that for domains, such as text, which afford many features as well as many potential feature interactions, having an algorithm which can explicitly model these feature dependencies efficiently, yet still produce guaranteed compact models is desirable in itself.

We now turn to the details of the KDB algorithm. Our algorithm is supplied with a dataset of training instances D as well as the k value for the maximum allowable degree of feature dependence. It begins with a Bayesian network BN that contains just the nodes representing the features and the class variable (and no arcs between them). The algorithm determines an ordering for the features based on their mutual information with the class. Then, using this feature ordering, the algorithm determines the set of up k parent nodes for each feature in the network. Once the structure of the network is determined, it is then possibly to compute the conditional probability tables needed at each node directly from the input data. Thus, the algorithm outputs a k-dependence Bayesian classifier, complete with conditional probability tables. We note that priors may be easily incorporated into the conditional probability tables of the classifier, but we omit them in the presentation here. The algorithm is as follows:

`Procedure KDB`

1. $\forall X_i$ `Compute` $\alpha_i \leftarrow MI(X_i; C)$
2. `Sort and renumber features` $X_1, \ldots, X_n$ `in descending order by` α_i
3. $\forall i, j \;\; i \neq j$ `Compute` $\gamma_{ij} \leftarrow MI(X_i; X_j \mid C)$
4. `For` $i = 1, \ldots, n$ `do`
    - 4.1. $r \leftarrow \min(i-1, k)$
    - 4.2. $\boldsymbol{X}_{d_i} \leftarrow r$ `features` $X_{j_1}, \ldots X_{j_r}$ `with largest` γ_{ij_ℓ}`, where` $j_\ell < i$
    - 4.3. `parents(`X_i`)` $\leftarrow C \cup \boldsymbol{X}_{d_i}$
5. `Compute the conditional probability tables inferred by the structure of the Bayesian network by using counts from dataset` D

Note that in Step 3 of the algorithm, for each pair of features X_i and X_j, we compute the class conditional mutual information between these features [35], denoted

$MI(X_i; X_j \mid C)$. This quantity is defined as

$$MI(X_i; X_j \mid C) = \sum_{x_i, x_j, c} \mathrm{P}(x_i, x_j, c) \log \frac{\mathrm{P}(x_i, x_j \mid c)}{\mathrm{P}(x_i \mid c) \cdot \mathrm{P}(x_j \mid c)} \tag{8.3}$$

Furthermore, in this description of the algorithm, Step 4.2 requires that the set of conditioning features $\boldsymbol{X}_{d_i}$ for each feature X_i being added to the model contain r elements. To make the algorithm more robust, we also consider a variant where we change this step to be:

4.2 $\boldsymbol{X}_{d_i} \leftarrow q$ `features` $X_{j_1}, \ldots X_{j_q}$ `with largest` γ_{ij_ℓ},
`where` $j_\ell < i$, $\gamma_{ij_\ell} > \theta$ `and` $q \leq r$

In this revised step, θ is a pre-set mutual information threshold which any given pair of features must exceed in order for a dependency to be added between these variables in the model. This threshold helps make the learning algorithm more robust in that it prevents the modeling of weak feature correlations that may appear simply due to poor probabilities estimated from limited data. It can also be interpreted as a form of prior over network structures which prefers sparseness—an instantiation of Occam's razor[13]. Thus, we can hope to attenuate the effect of modeling spurious dependencies when the value of k is set too high. In doing so, we can also reduce the number of parameters that must be estimated in the resulting model, and hence try to control the overall variance associated with the classifier. This is especially important for text domains where the large number of features may make spurious correlations more common.

Another feature of our algorithm which makes it very suitable for text domains is its relatively small computational complexity. Computing the actual network structure requires $O(n^2(mcv^2 + k))$ time, where n is the number of features, m is the number of data instances, c is the number of classes, and v is the maximum number of discrete values that a feature may take. The first term in this complexity result comes from computing the n^2 mutual information values in Step 3, each of which requires $O(mcv^2)$ time to compute. The second term comes from the loop in Step 4, where $O(nk)$ time is required to determine the (at most) k parents of each of n feature

Dataset	No. Classes	No. Features	Training Set Size	Testing Set Size
Corral	2	6	128	10-fold CV
LED7	10	7	3200	10-fold CV
Chess	2	36	3196	10-fold CV
DNA	3	180*	3186	10-fold CV
Vote	2	48*	435	10-fold CV

Table 8.1: Datasets from the UCI repository used in the initial experiments with KDB. *Reflects Boolean encoding of feature values.

nodes. Once the network structure has been determined, calculating the conditional probability tables within the network (Step 5) takes $O(n(m + cv^k))$ time since, for each of n feature nodes, we must make a pass through all m data instances to compute the cv^k entries in the corresponding conditional probability table. In many domains, v will be small (for example, in our binary encoding of text documents $v = 2$) and k is a user-set parameter, so the algorithm will scale linearly with m, the amount of data in the dataset D. Moreover, classifying an instance using the learned model only requires $O(nck)$ time.

8.4 Initial Results on Non-Text Domains

As in our work in the previous chapter, we first tested the KDB algorithm on several well-understood datasets from the UC Irvine repository. This allowed us to get an initial feel for the strengths and limitations of our algorithm prior to returning to our primary focus of text classification. These datasets are described in Table 8.1. Largely, these datasets are the same ones used in our feature selection work.

Specifically, in these experiments we wanted to measure if increasing the value of k above 0 would help the predictive accuracy of the induced models. In other words, we sought to compare the dependency modeling capabilities of KDB with Naive Bayes. Moreover, we wanted to see if we could uncover various levels of dependencies that we know exist in a few artificial domains by seeing how classification accuracy varied with the value of k. We also tested both the original and modified KDB algorithm

Dataset	k	Accuracy KDB-original	Accuracy KDB-θ
Corral	0	$88.4 \pm 10.5\%$	$88.4 \pm 10.5\%$
	1	**100.0 ± 0.0%**	**100.0 ± 0.0%**
	2	$96.7 \pm 5.8\%$	$96.7 \pm 5.8\%$
	3	$88.4 \pm 17.2\%$	**100.0 ± 0.0%**
LED7	0	$72.9 \pm 2.1\%$	$72.9 \pm 2.1\%$
	1	$73.1 \pm 3.9\%$	$73.0 \pm 2.9\%$
	2	**73.5 ± 2.3%**	$72.9 \pm 2.4\%$
	3	$73.2 \pm 2.3\%$	**73.4 ± 1.3%**
Chess	0	$86.2 \pm 1.9\%$	$86.2 \pm 1.9\%$
	1	$93.9 \pm 1.3\%$	$93.8 \pm 1.4\%$
	2	**95.1 ± 1.2%**	**95.5 ± 1.6%**
	3	$94.9 \pm 1.1\%$	$95.3 \pm 1.2\%$
DNA	0	$94.0 \pm 0.9\%$	$94.0 \pm 0.9\%$
	1	$94.0 \pm 1.6\%$	$94.1 \pm 1.1\%$
	2	**95.3 ± 1.2%**	**95.6 ± 1.1%**
	3	$93.3 \pm 0.9\%$	$95.5 \pm 1.8\%$
Vote	0	$90.2 \pm 3.8\%$	$90.2 \pm 3.8\%$
	1	$92.6 \pm 3.6\%$	$92.1 \pm 5.3\%$
	2	$92.3 \pm 3.5\%$	$93.5 \pm 4.1\%$
	3	**93.0 ± 2.2%**	**94.0 ± 3.2%**

Table 8.2: Classification accuracies for KDB on UCI datasets.

(which employs the mutual information threshold, θ). For these experiments, we set $\theta = 0.03$, which was a value heuristically determined from very preliminary runs of the algorithm. Nevertheless, we note that in a more rigorous usage scenario, it would be straightforward (and desirable) to optimize this parameter using a wrapper approach [92]. The results of our experiments on the UCI datasets are given in Table 8.2 with KDB-original referring to the original algorithm and KDB-θ referring to the variant using the mutual information threshold.

The two artificial domains, Corral and LED7, were selected because of known dependence properties. As explained previously, the Corral dataset represents the concept $(A \wedge B) \vee (C \wedge D)$ and thus is best modeled when at least one feature dependency is allowed, as is borne out in our experimental results (higher accuracies when $k > 0$). The LED7 dataset, on the other hand, actually satisfies the Naive Bayes assumption, with all the features being independent when conditioned on the class. Our algorithm helps discover this phenomenon, as reflected by the similar accuracy rates when $k = 0$ and $k > 0$. Note that the classification accuracies in the $k = 0$ and $k > 0$ are not identical due to the fact that this dataset contains artificial noise which can contribute to higher variance in parameter estimation.

In the real-world domains from UCI, we find that modeling feature dependencies very often improves classification results. This is especially true for the KDB-θ algorithm, where classification accuracies when $k > 0$ are almost always greater than or equal to the $k = 0$ (Naive Bayes) case. In the Chess ($k = 1, 2, 3$), Vote ($k = 2, 3$) and DNA ($k = 2, 3$) domains, these improvements are statistically significant (t-test with $p < 0.10$). Moreover, by noting how the classification accuracy changes with the value of k we get a notion of how to set the parameter k for each problem and get a rough approximation to the degree of feature dependence in each domain. For example, in both the Chess and DNA datasets, we see large jumps in accuracy when going from $k = 0$ to $k = 2$, and that there is no gain when $k = 3$. This indicates that there are either many low-order interactions in these domains, or that the increase in variance of the model parameters resulting from data fragmentation causes no further gain in accuracy to be realized from modeling the additional dependencies. (We return to this latter point below.) In either case, it is easy to identify that 2-dependence models provide the best results for both of these domains.

It is important to note that the Boolean encoding of the Vote and DNA domains has introduced some feature dependencies into the data, but such representational issues (which are often unknown to the end user of a classification system) also argue in favor of methods that can model such dependencies when they exist. Recall that as k grows, we must estimate a larger probability space (more conditioning variables) with the same amount of data. This growth in the number of conditioning features can cause our probability estimates to become more inaccurate since each is based on fewer data instances, and thus leads to an overall decrease in predictive accuracy. This phenomenon is reflected in the difference between using $k = 2$ and $k = 3$ in many

Dataset	Initial feature set size	Reduced feature set size
D1	1143	50
D2	1001	50
D3	552	20
D4	1126	50
D5	1953	100

Table 8.3: Reduced feature set sizes of Reuters datasets.

of the domains. The KDB-θ algorithm is less prone to this effect, but it is still not impervious. Still, our initial results indicate that we can induce better probabilistic classification models for domains, such as text, that contain feature dependencies.

8.5 Results on Text Domains

Returning, once again, to our original problem of text classification, we now examine the efficacy of the KDB algorithm on Reuters subsets D1 through D5, which we have used throughout our work. For these experiments, we began by employing the feature selection algorithm described in the previous chapter to reduce the feature set for each dataset to approximately 5% of its initial size. This level of reduction was chosen since it seemed to provide good classification results with Naive Bayes in our earlier work. When employing the feature selection algorithm, we used Markov blanket (abbreviated MB below) sizes of both 0 and 1. The actual reduction in feature set size is summarized in Table 8.3.

Furthermore, since our initial results with KDB showed that the variant which incorporated the mutual information threshold θ generally provided slightly better results than the original algorithm, we use that variant in all our runs on the Reuters data. As before, we simply set $\theta = 0.03$ in these experiments. The average accuracy results (as well as standard deviations) using 10-fold cross-validation in these experiments is reported in Table 8.4. Note that we applied KDB (with $k = 0, 1$, and 2) *and* feature selection to each fold so as not to give the feature selection algorithm access

Dataset	k	Feature selection MB size = 0	Feature selection MB size = 1
D1 (50 features)	0	99.2 ± 1.1%	99.4 ± 1.0%
	1	99.2 ± 1.5%	99.2 ± 1.5%
	2	99.0 ± 1.5%	98.8 ± 1.8%
D2 (50 features)	0	97.8 ± 2.5%	98.3 ± 2.7%
	1	98.9 ± 1.5%	98.7 ± 1.5%
	2	98.3 ± 1.7%	98.5 ± 1.5%
D3 (20 features)	0	93.9 ± 4.8%	94.3 ± 4.5%
	1	95.7 ± 3.7%	93.9 ± 2.9%
	2	94.3 ± 5.1%	94.3 ± 4.5%
D4 (50 features)	0	97.2 ± 2.3%	97.0 ± 2.1%
	1	95.7 ± 2.0%	95.7 ± 2.3%
	2	95.2 ± 2.5%	95.4 ± 2.8%
D5 (100 features)	0	78.5 ± 2.0%	74.4 ± 4.2%
	1	83.1 ± 2.5%	80.6 ± 3.5%
	2	83.2 ± 3.1%	81.9 ± 3.6%

Table 8.4: Classification accuracies for KDB on text datasets.

to the testing data.

From these results, we initially note that KDB with $k = 1$ or 2 generally performs slightly better when the feature selection algorithm previously applied to the data uses an a Markov blanket of size 0 (i.e., the feature selection algorithm does not make use of conditioning to eliminate correlated features). This makes sense given that the KDB algorithm is aimed at capturing such dependencies when they exist and thus its expressivity can account for such correlated features. Moreover, since running the feature selection algorithm without conditioning information is much faster than when such information is incorporated, it becomes very computationally efficient to run the feature selection algorithm in this fashion and then apply KDB to model dependencies in the reduced feature space.

While all the methods appear to be performing quite well, it is important to note that modeling dependencies (using $k = 1$ or 2) in these text datasets never significantly degrades performance over Naive Bayes (using $k = 0$), but can lead to significant accuracy gains in some cases. For example, on both datasets D2 and D3 using $k = 1$ (and a Markov blanket size of 0 during feature selection) leads to significant improvements over Naive Bayes (paired t-test $P < 0.10$). The most dramatic increase is seen on the most difficult and largest dataset D5, where using either $k = 1$ or 2 *always* outperforms Naive Bayes by a significant margin ($P < 0.01$). In dataset D4, we actually see a slight degradation over Naive Bayes, but these results are not significant. Hence, we believe that the combination of feature selection and the KDB algorithm can be very effective in text domains.

To provide further experimental evidence for the utility of dependency modeling with KDB, we considered the task of classifying the complete updated Reuters-21578 dataset.[1] This dataset contains 118 topics (classes) and (using the "ModApte" split of the data) has 9603 training documents and 3299 testing documents. Each document, however, may be labeled with multiple topics. Consequently, we follow the traditional protocol for this collection of treating this problem as 118 separate binary classification tasks. Each such task corresponds to predicting whether or not documents belong in a given topic.

We began by performing a series of preliminary experiments using *only* the training data (which was split into a 7147 document training set and a 2456 document validation set). First, we sought to determine how many features we should use by applying our feature selection algorithm to this data. These results indicated that using our feature selection algorithm (with Markov blanket size set to 0 for computational efficiency[2]) led to the best performance for both Naive Bayes and KDB when the feature set for each of the binary classification tasks was reduced to 50 features. Moreover, our preliminary experiments also indicated that using KDB with $k = 2$ generally outperformed KDB with $k = 1$. Consequently, we only considered

[1]The results on the Reuters-21578 collection were obtained in collaboration with Dumais, Heckerman and Platt [48]. In that work, we also present results on this dataset using other (non-Bayesian) algorithms, but do not include those results here for brevity (since they are not germane to our main discussion). We also thank David Lewis for making the Reuters-21578 collection publicly available. More information on this dataset is available at `http://www.research.att.com/~lewis/reuters21578.html`.

[2]Recall that using our feature selection algorithm with a Markov blanket size of 0 is equivalent to selecting the features which have the highest mutual information with the class variable.

Topic	Naive Bayes	KDB with $k = 2$
earn	95.9%	95.8%
acq	87.8%	88.3%
money-fx	56.6%	58.8%
grain	78.8%	81.4%
crude	79.5%	79.6%
trade	63.9%	69.0%
interest	64.9%	71.3%
ship	85.4%	84.4%
wheat	69.7%	82.7%
corn	65.3%	76.4%
Avg. top 10 categories	81.5%	85.0%
Avg. all categories	75.2%	80.0%

Table 8.5: Breakeven points on full Reuters dataset.

the former in our final experiments.

With these experimental settings in hand, we then tackled the full Reuters-21578 dataset. For each binary classification task, our feature selection algorithm was applied to the full training set to reduce the feature space to 50 features. We then trained our classifiers (Naive Bayes and KDB with $k = 2$) on this data and subsequently used them to classify the testing data. We scored the classification results for each algorithm using the traditional score for this task of the *breakeven point*, as opposed to the classification accuracy. The breakeven point is defined to be the average of the *precision* and *recall* for each topic. Precision is defined as the the number of documents correctly classified into a topic divided by the total number of documents classified into that topic by the classifier. Recall is the number of documents correctly classified into a topic divided by the total number of documents which truly belong in that topic. The results of these final experiments on the full collection are given in Table 8.5.

We report the breakeven points for each of the 10 topics in the collection which contain the most documents. We also give the micro-averaged (i.e., weighted average by category size) breakeven points for the the top 10 categories as well as the entire collection (all 118 categories). As can be seen in both the results from the individual topics as well as the aggregated results, using KDB to learn limited dependence Bayesian classifiers leads to consistent improvements in accuracy over using the simple Naive Bayesian model. This matches our intuitions about text domains, since we would expect them to contain many dependencies between words.

8.6 Conclusions and Related Work

We have defined a spectrum of dependency for Bayesian classification algorithms and presented an efficient algorithm, KDB, that allows for traversal along this spectrum. The algorithm was analyzed both in terms of its theoretical complexity and empirical performance. The empirical results show that we can use such an algorithm not only to help discover dependencies between domain features, but also to help improve classification accuracy as a result. We find that this algorithm is generally effective at producing better models of text domains, where our intuitions would lead us to believe that many feature dependencies would exist. In these cases, the algorithm seems to work well when used in conjunction with feature selection. We further explore the coupling of feature selection and KDB in the context of hierarchical classification in the next chapter.

It is important to point out that many research efforts have focused specifically on ways to increase the expressivity of the Naive Bayesian classifier. While we discussed those algorithms most closely related to KDB in Section 8.3 to more clearly contrast them with our work, we present other (different) methods here. One of the earliest such attempts is Kononenko's work on "Semi-Naive" Bayesian classifiers [96] which considered creating new features from pairs of existing features as a means of relaxing the strict independence assumption. Unfortunately, this method was not capable of realizing any empirical accuracy improvements over Naive Bayes.

Similarly, Pazzani [124] has looked at constructive methods to capture feature dependencies in Naive Bayes by creating "higher order" features that are simply combinations of several simple features. This approach relies heavily on searching for useful feature combinations by using classification accuracy as a search metric (akin to the wrapper approach). As a result, the method can have difficulty while searching in high-dimensional feature spaces due to local minima and must rely on generating precise accuracy estimates by performing an expensive cross-validation operation at each step in the search. This difficulty can make the algorithm impractical for use in text domains.

The work of Heckerman, Geiger and Chickering [72] has been successful at casting the problem of learning Bayesian network structure into an entirely Bayesian framework, and is similar in some respects to the K2 algorithm described earlier. They provide a great deal of theoretical analysis for their methods, and, among other things, eliminate the need for an initial ordering over domain features (as is required by K2). Similar to K2, however, classification was not the primary concern of their work, as they were focused on general density estimation problems. This difference in focus may render their induced networks less suitable for the classification task.

Other researchers have sought to extend K2 to deal more specifically with classification problems. For example, Singh and Provan [152] use an algorithm named CB [154], which is heavily based on K2, for determining the structure of a Bayesian classifier. However, their algorithm disregards the fact that they are trying to optimize the Bayesian network solely for predicting the class variable, and instead optimizes the network structure to model the whole joint probability of the data. In other words, rather than building a structure specifically to model $\mathrm{P}(C \mid \boldsymbol{X})$, the algorithm optimizes the Bayesian network structure to model $\mathrm{P}(C, \boldsymbol{X})$, and can thus spend much time modeling dependencies between features which are not really relevant to the classification task at hand.

Our algorithm also has much room for further work. Ways of identifying good values for k for a given domain need to be determined (possibly using cross-validation). Also, the value of the θ parameter needs a more solid foundation, so that this parameter may be set automatically based on the the characteristics of a given domain. Finally, we would like to find ways of extending our algorithm to sit more firmly within the framework of Bayesian learning of network structures.

Chapter 9

Hierarchical Classification

9.1 Introduction

In tackling the problem of document classification, it is important to utilize the organizational schemes that are often imposed on such information. Historically, one of the most successful paradigms for organizing large quantities of text (thereby making the topical contents of a collection more comprehensible to end users), is to categorize documents according to their topics, where these topics are organized in a hierarchy of increasing specificity. For example, hierarchical classifications of this type have long been used in special-purpose collections of documents such as the MEDLINE collection of medical literature [76] or collections of patent documents [150]. More generally, the classification schemes used by many libraries (both the traditional and new "digital" varieties) rely on hierarchical structures. Such hierarchical organizational schemes have also been used by Internet "portal" sites, such as *Yahoo!* [173] and *Infoseek* [78], in an attempt to categorize the contents of the World Wide Web. And even at the level of the individual user, Web page bookmarks are often stored in hierarchies (akin to the hierarchical file system structure used on most personal computers today). Indeed, part of our motivation for building SONIA was to enable users to quickly and easily generate and maintain such topical hierarchies of documents.

As alluded to in the first chapter of this work, the unfortunate bottleneck traditionally encountered in classifying documents into such organizational schemes is the need for a person to manually read each document and decide on its appropriate place in the hierarchy. Clearly, we would like systems such as SONIA to help users alleviate this bottleneck by automatically classifying new documents for them. In fact, Infoseek has recently attempted to overcome this difficulty by using neural network technology to automatically categorize Web pages [79]. In many ways, this task is ideally suited to the application of machine learning techniques. We have a specified set of classes (i.e., the topics in the hierarchy), and a very large training set, consisting of all of the documents that have already been classified. However, with few exceptions (notably [4] which focused on hierarchically structured attributes rather than classes), most work in classification has ignored the problem of supervised learning in the presence of hierarchically structured classes.

Of course, standard classification techniques, such as the one outlined in the previous chapter, can be applied to this problem almost directly. We simply construct a "flattened" class space, with one class for every leaf in the hierarchy. We can now train a single classifier so that each document is classified as belonging to precisely one of the possible basic classes.[1] Unfortunately, this simplistic approach can be problematic in the context of text classification. Here, the resulting classification problem may be huge. For example, even a moderately large corpus may contain hundreds of classes and many thousands of features. The computational cost of training a classifier for a problem of this size is prohibitive. Furthermore, the variance of the resulting classifier is typically very large, since such a model will have many thousand parameters that need to be estimated, and thus can easily lead to overfitting of the training data. As a result of the potentially enormous growth in the parameter space, we are typically able to use only very simple classifiers such as Naive Bayes.

Previous work [148, 94, 175], has shown that feature selection can be a useful tool in dealing with these issues. We can simply eliminate many of the words that appear in the corpus as being unindicative of any topic. In Chapter 7, for example, we showed that in many cases one can obtain a significant increase in accuracy by reducing the number of words used for classification down to as little as 5% of the original feature

[1] Here, we focus on disjoint topic hierarchies. Similar techniques can be applied for the non-disjoint case.

set. Still, even such an extreme reduction may yield several hundred features for a large collection. In such cases, computational cost and issues of robustness still pose significant limitations for all but the simplest classification algorithms.

We propose a new approach to classification using a structured hierarchy of topics. Rather than ignoring the topical structure and building a single huge classifier for the entire task, we use the hierarchical structure to break the problem up into manageable size pieces. The basic insight supporting our approach is that topics that are close to each other in the hierarchy typically have a lot more in common with each other than topics that are far apart. Therefore, even when it is difficult to find the precise topic of a document (e.g., "color printers"), it may be easy to decide whether it is about "agriculture" or about "computers".

Building on this intuition, our approach divides the classification task into a set of smaller classification problems corresponding to the splits in the classification hierarchy. Thus, for example, we may have one classifier which distinguishes articles about agriculture from articles about computers, and another one, only applied to documents about agriculture, which distinguishes animal husbandry from crop farming. Each of these subtasks is significantly simpler than the original task, since the classifier at a node in the hierarchy need only distinguish between a small number of categories. Therefore, it is possible to make this determination based only on a small set of features. For example, there appears to be a fairly small number of words—e.g., *computer, farm, plant, software, ...*—whose presence or absence in the document clearly differentiates documents about agriculture from documents about computers. The ability to restrict to a very small feature set avoids many of the difficulties we describe above. The resulting models are more robust, and less subject to overfitting.

It is important to note that the key here is not merely the use of feature selection, but its integration with the hierarchical structure. To understand this integration, observe that the set of features required for these subtasks varies widely from one to the other. For example, almost none of the words that can help us differentiate between agriculture and computers are useful for distinguishing between animal husbandry and crop farming: a word such as "farm" is unlikely to be helpful because it is fairly likely to appear in documents of both types, whereas a word such as "computer" is not helpful because it is likely to appear in virtually no documents that reach this classifier. Thus, while each classifier uses only a very small set of features, the overall set of features used in the classification process is still rather large. A single flattened classifier would have to consider all of these features in order to do a reasonable job of classifying all of the documents. For any given document, however, most of these features are irrelevant, and serve only to confuse the classifier. In the hierarchical approach, a document percolating down the hierarchy of classifiers only encounters questions concerning a small fraction of the features throughout the process (e.g., a document about computers will probably never meet a classifier utilizing the word "cow"). Even the features which each classifier does utilize are divided so as to focus the attention of each classifier on the features relevant to the classification subtask at hand.

The reduction in the feature space also allows us to go beyond the simple classifiers such as Naive Bayes, to which we are restricted in tasks involving a very large number of features. For example, we can train a probabilistic classifier, such as KDB described in Chapter 8, that takes into account the correlation between different features (e.g., the fact that *Microsoft* and *Windows* tend to co-occur). Such classifiers provide a more realistic model for text data, often leading to higher classification accuracies. However, the search in this more complex hypothesis space is significantly more expensive, being at least quadratic in the number of features for an algorithm such as KDB (and potentially exponential if we tried to learn *optimal* probabilistic classifiers), versus the linear time behavior of Naive Bayes. Furthermore, even when we are willing to spend the time, these more complex models are usually unsuitable for large feature spaces, due to the problem of overfitting.

As in the case of feature selection, it is the integration of this idea with the hierarchical structure which is the key to its success. As our results show, a flat classifier has much more difficulty taking advantage of the richer models, regardless of the size of the feature space. By contrast, when we use more expressive classifiers in the nodes of our hierarchy, the classification accuracy increases much more substantially. We conjecture that, as for the choice of feature set, the dependency models appropriate for the different nodes in the hierarchy are radically different. In the flat classification case, the dependency model would be an aggregate of the various lower-level models. As a consequence, in corpora with many topics, the dependencies may either "wash out" or be too complex to represent adequately using a limited dependency model.

The use of the hierarchical structure allows us to focus both our feature space and the dependency model on the relevant distinctions. As we show, the combination of these three techniques provides significant accuracy gains over the standard flat approach.

We note that, while our techniques are originally motivated by the large classification tasks arising in the context of text domains (and more specifically, the SONIA system), they may also be useful in other domains. For example, in medical applications, we often want to classify a patient's disease based on symptoms and test results. Here also, our classes—the diseases—are often organized in a taxonomic hierarchy, where only a small number of features is needed to distinguish between neighboring classes.

The rest of this chapter is structured as follows. In Section 9.2 we briefly review the specific details of how we use our feature selection and classification algorithms in a hierarchical context. While we focus on the probabilistic techniques presented in previous chapters, we emphasize that our basic paradigm for hierarchical classification in no way depends on the use of these particular techniques. In Section 9.3, we provide a variety of experimental results supporting our approach. We show that our technique allows us to restrict the set of features significantly (e.g., from over 1000 to 10), and that the resulting classifier, in addition to being smaller and easier to train, also provides better accuracy than the flat classifier. Section 9.4 then presents some theoretical results on extending the use of our hierarchical classification scheme to deal with classes organized into directed acyclic graphs (as opposed to strict tree structures). We conclude in Section 9.5 with some discussion and directions for future work.

9.2 Hierarchical Classification Scheme

Our general approach, as described in the introduction, consists of constructing a hierarchical set of classifiers, each based on its own set of relevant features. It uses two main subroutines: a feature selection algorithm for deciding on the appropriate feature set at each decision point, and a supervised learning algorithm for constructing a classifier for that decision. The general approach can be instantiated in a variety of ways, depending on the choice of these subroutines.

Here, we have chosen to focus on the probabilistic methods for feature selection and for classification given in Chapters 7 and 8. These methods provide both efficient and principled techniques for pruning large feature sets as well as ways to induce a range of classifiers of varying complexities (and accuracies). We now provide a brief overview of how these methods are applied within our hierarchical classification scheme.

9.2.1 Feature Selection

As mentioned throughout this work, our representation of text documents as vectors leads to a very large number of initial features. Consequently, even if we use algorithms such as KDB, which are "merely" quadratic (as opposed to exponential) in the total number of features, the computational cost can still be very great. Moreover, if we wish to employ more optimal models which are of exponential complexity in the size of the feature set, then feature selection is an absolute necessity.

In addressing this issue, we apply the feature selection method described in Chapter 7. To review, the algorithm greedily eliminates features one by one so as to least disrupt the original conditional class distribution $\mathrm{P}(C \mid \boldsymbol{X})$. For each remaining feature X_i, the algorithm finds a feature set MB_i which attempts to approximate the Markov blanket of X_i (i.e., the set of features that render X_i conditionally independent of the remaining domain features). The algorithm then determines the expected relative entropy, given by

$$\Delta_i = \sum_{X_i, MB_i} \mathrm{P}(X_i, MB_i) \cdot D(\mathrm{P}(C \mid X_i, MB_i), \mathrm{P}(C \mid MB_i)) \tag{9.1}$$

and eliminates the feature X_i for which Δ_i is minimized. This process can be iterated to eliminate as many features as desired.

In the hierarchical classification work described here, we compute $\mathrm{P}(C \mid X_i, MB_i)$ in the feature selection algorithm by making the simplifying assumption that $\forall i\ MB_i = \emptyset$. This allows the algorithm to be run quickly (in $O(mn \log n)$ time, where m is the number of documents and n is the number of features), even on many thousands of features. In this respect, the algorithm is even more applicable to text domains with many features. Moreover, as our results in the previous chapter show, assuming $MB_i = \emptyset$ during feature selection does not pose significant problems for subsequently applying induction algorithms that can actually model the dependencies that exist between the remaining features. Again, we note that using our feature selection algorithm with $MB_i = \emptyset$ is equivalent to selecting the feature that have highest mutual information with the class variable.

9.2.2 Bayesian Classification

In our hierarchical classification scheme[2] we begin by applying probabilistic feature selection to the entire training dataset, using, at first, just the topics associated with each document at the first tier in the hierarchy as classes. The resulting reduced feature set is then used to build a probabilistic classifier for the first tier of the hierarchy. We currently consider Naive Bayes in addition to KDB with $k = 1$ and $k = 2$ as our classification methods. Then, for the training documents in each of the first-tier topics, the second-tier topics are used as class labels. For each first-tier topic, a separate round of probabilistic feature selection is employed. Note that this feature selection is done starting from the original feature set (pruned only based on

[2]For the time being, we assume that topic hierarchies are tree structured. Later, in Section 9.4, we show how this condition may be relaxed.

Zipf's law as described in Chapter 2), since, as we have observed, the most indicative features at one level of the hierarchy are unlikely to be particularly useful at lower levels. We construct a separate classifier for the subtopics of each first-tier topic using the appropriate reduced feature set.

Although we only consider two level hierarchies in our empirical results, it is straightforward to extend this approach to hierarchies of arbitrary depth. We do so as follows: at each node of the hierarchy we simply perform feature selection with respect to the classes defined by the sub-topics of this node. Using this reduced feature set, we then build a classifier which can filter new instances down to the next level of the hierarchy. This process of feature selection and classifier induction is repeated at each node down through the hierarchy, with the hierarchy leaves representing the final classes which documents may be placed into. Moreover, since every node in the hierarchy has only a subset of the total class labels, and the nodes at the lower tiers of the hierarchy have fewer instances each, the additional cost of feature selection and induction is not substantially more than that of the flat classification scheme.

Test documents are classified by filtering them through the hierarchy of classifiers. First, the top-level classifier sends the document down a particular branch to another classifier at the next level of the hierarchy. This process is repeated until the document reaches a leaf node in the hierarchy and a final class assignment is made. Due to the series of classification decisions that are made in this scheme, we colloquially refer to it as the "Pachinko machine" classifier.[3] Note that errors made at the higher levels of the hierarchy are unrecoverable at lower levels. Thus, in the two-level hierarchies we consider here, our method needs to make two correct classifications in order for a test document to be considered properly classified. We have conducted some initial experiments in which documents were sent down multiple paths in the hierarchy of classifiers as a way of addressing this issue, but, surprisingly, did not obtain improved classification results.

In the flat classification scheme, we simply treat every low-level topic (leaf node)

[3]A Pachinko machine is a Japanese game in which small steel balls are dropped into a vertical playing area filled with hundreds of small nails that direct the flow of the balls downward. The goal of the game is to get as many balls as possible into special scoring slots at the bottom of the playing area.

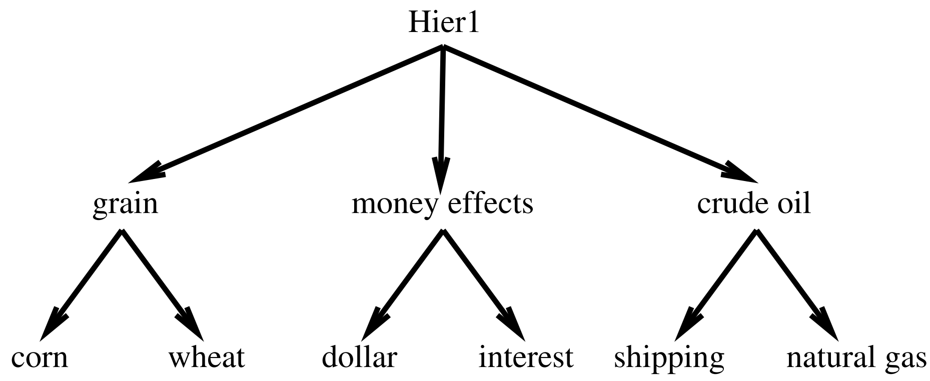

Figure 9.1: Hier1 hierarchy, containing 939 documents and 1568 features.

as a separate class. We then apply probabilisitic feature selection and induce one probabilistic model based on this reduced feature set.

In order to induce classification models that more accurately capture the distribution of word appearances in text, we make use of the limited dependence Bayesian classifers introduced in the previous chapter. Our experiments there revealed that generally using 1 and 2-dependence models generally seemed to provide the best results, as data fragmentation problems were seen when trying to induce models with even greater degrees of dependence. We subsequently refer to these models as KDB-1 and KDB-2, respectively. For comparative purposes, we also employ the Naive Bayesian classifier in our hierarchical scheme.

9.3 Results

In order to test our scheme for hierarchical classification, we first needed to obtain hierarchically classified text data. In keeping with our previous experimental evaluations, we used the Reuters-22173 dataset. The Reuters collection does not have a pre-determined hierarchical classification scheme, but each document can have multiple labels. We therefore identified labels which tended to subsume other labels, and used those as the higher level topics in our hierarchy. Three hierarchical subsets of the Reuters collection, which we call Hier1, Hier2 and Hier3, were then extracted. These datasets are described in Figures 9.1, 9.2 and 9.3, respectively.

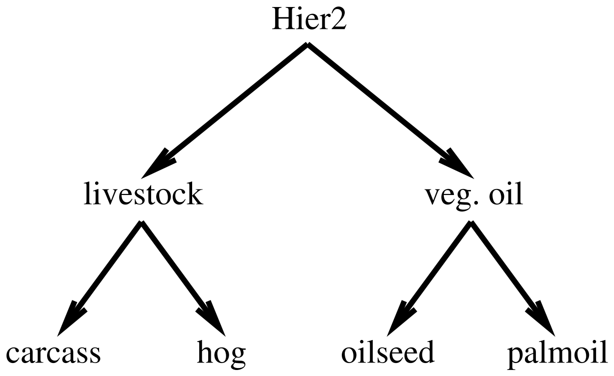

Figure 9.2: Hier2 hierarchy, containing 138 documents and 435 features.

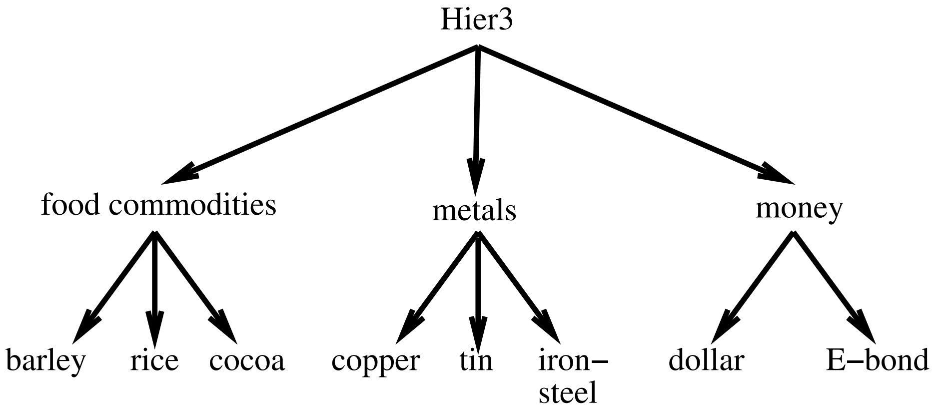

Figure 9.3: Hier3 hierarchy, containing 834 documents and 1440 features.

As dicussed in Chapter 2, we applied a single pass of the Zipf's law-based feature selection method to eliminate all words which appeared fewer than 10 or more than 1000 times in each corpus. Recall that we also did this in our initial work on feature selection, so that we would not get unrealistic improvements in accuracy by simply eliminating features that rarely if ever appear during testing, or are so frequent that they will have no bearing on classification. The number of features reported for each dataset is *after* the application of this initial feature selection. These datasets were then used in our experiments, detailed below, employing 10-fold cross-validation to produce multiple training and testing sets for each dataset.

In our experimental work, we seek to show that the hierarchical approach compares favorably with the simple approach of constructing a single large classifier over a flattened topic space. In both cases, the feature selection phase plays a crucial role in the performance of the resulting classifier. Since it is our belief that a small set of features suffices for accurately distinguishing between topics (and furthermore helps avoid overfitting), we employed a very aggressive feature selection policy. In the Hier1 and Hier2 domains we considered reducing the feature space to 10, 40, and 160 features. For Hier3, which contains the largest number of final classes among which we need to distinguish, we considered 20, 80, and 320 features. We chose this particular pattern of feature space sizes in order to facilitate a fair comparison between the flat method and the hierarchical method. Recall that, in the hierarchical case, a potentially very different set of features is selected at each node in the hierarchy. Therefore, the hierarchical method, as a whole, actually examines a larger set of features. To allow a fair comparison, we also compare the hierarchical method with some number of features to a flat method with four times as many features (since our hierarchies contain around four classifiers each). We also considered running each method on the full feature set, but since the number of features was very large, it became prohibitive to consider models which did not assume conditional independence of features (e.g., KDB); thus only the results for Naive Bayes are given in these cases. The accuracies and standard deviations for 10-fold cross-validation using the hierarchical approach are given in Table 9.1. The analogous results in the flat classification case are shown in Table 9.2.

Dataset	# Features	Hierarchical NB	KDB-1	KDB-2
Hier1	10	92.4 ± 2.4	**92.4 ± 2.3**	92.2 ± 2.5
	40	91.8 ± 2.8	93.3 ± 1.9	**94.1 ± 2.7**
	160	88.2 ± 2.3	**91.2 ± 2.8**	91.1 ± 3.0
	1568	85.9 ± 3.1	—	—
Hier2	10	**87.7 ± 7.4**	85.4 ± 8.5	85.4 ± 8.5
	40	88.5 ± 9.1	87.7 ± 8.3	**90.0 ± 8.2**
	160	86.2 ± 10.8	**86.2 ± 10.1**	82.3 ± 10.9
	435	84.6 ± 9.6	—	—
Hier3	20	90.1 ± 4.1	94.7 ± 3.0	**94.9 ± 2.6**
	80	90.9 ± 2.4	**98.6 ± 1.4**	97.7 ± 1.8
	320	94.8 ± 2.5	**95.5 ± 2.0**	94.2 ± 2.5
	1440	92.8 ± 2.5	—	—

Table 9.1: Accuracy percentages for hierarchical learning employing feature selection.

We begin by noting the substantial improvements in accuracy when feature selection was aggressively employed versus the case where all domain features were used. We see an improvement both in the hierarchical case and in the flat case, and for every single dataset. In the Hier1 dataset, these gains are particularly visible, with a statistically significant difference (t-test with $P < 0.01$) obtained in Naive Bayes for both the flat classifier with 40 features (versus the original 1568) and the hierarchical classifier with 10 features. In both cases, around 1500 features—over 95% of the features *after* Zipf's law pruning—were eliminated. Of course, the number of features cannot be reduced to zero. In general, we observe an initial significant improvement as the number of features is reduced, and then a decrease in accuracy as the number of features is reduced too much. This phenomenon is yet another instance of the familiar bias-variance tradeoff, also seen in previous chapters. The selection of very few features, say 10, helps reduce the error associated with the variance of the model, but strongly biases the model. Conversely, the use of too many features, say 160, can cause the probability estimates used in the classification models to become inaccurate and thus lead to poorer overall performance. We believe that combining the

Dataset	# Features	Flat NB	Flat KDB-1	Flat KDB-2
Hier1	10	**79.3 ± 9.1**	79.1 ± 8.8	78.8 ± 8.7
	40	92.0 ± 3.1	**94.1 ± 2.3**	93.8 ± 2.6
	160	92.0 ± 4.1	**93.8 ± 2.8**	93.8 ± 2.8
	1568	86.2 ± 2.8	—	—
Hier2	10	**89.2 ± 8.3**	89.2 ± 8.3	86.9 ± 8.9
	40	**87.7 ± 5.4**	86.2 ± 6.1	86.2 ± 7.1
	160	**85.4 ± 8.5**	82.3 ± 12.6	77.7 ± 9.2
	435	83.1 ± 8.7	—	—
Hier3	20	**90.6 ± 3.4**	89.8 ± 2.4	89.8 ± 2.7
	80	95.4 ± 1.7	**95.7 ± 1.5**	95.2 ± 2.7
	320	**91.8 ± 4.1**	91.2 ± 2.7	89.3 ± 3.4
	1440	90.9 ± 2.5	—	—

Table 9.2: Accuracy percentages for flat learning employing feature selection.

hierarchical method with relatively aggressive feature selection can help address this issue since it allows a larger space of *overall* features to be considered, but provides a variance control for each classifier in the hierarchy by having it focus on just a few relevant features.

We now turn our attention to the main question: the difference between the hierarchical and flat classification methods. We begin by comparing the two approaches with an equivalent number of features for each classifier. (E.g., we compare the hierarchical classifier with 10 features in each node with the flat classifier with 10 features total.) In the case of Naive Bayes, the comparison is inconclusive. In some cases (Hier1 with 10 features) the hierarchical approach is significantly better, while in others (Hier3 with 80 features) the flat approach wins. Alternatively, we can compare the cases where both classification schemes see a roughly comparable number of features (i.e., hierarchical with 10 features per node versus flat with 40 features total). In this case, the comparison for Naive Bayes is still predominantly inconclusive (with one exception in Hier3 where the flat classifier wins).

Thus, the hierarchical approach appears to present few benefits when we restrict attention to simple classifiers such as Naive Bayes. However, as we explained in the introduction of this chapter, one of the primary benefits of the hierarchical approach is that it allows us to train more complex classifiers with richer dependency models. As we have seen, the complexity of algorithms for learning expressive classifiers grows rapidly with the number of features. Even when the growth is quadratic, as in the KDB algorithm, it is significantly more expensive to learn a single classifier over 160 features than to learn a few classifiers over only 40 features each. Furthermore, if we wish to construct even more accurate models by using an optimal Bayesian network learning algorithm, this task may be achievable in the case of 40 features, but is clearly infeasible in the case of 160.

Indeed, when we use these richer dependency models within the hierarchical approach, we begin to see significant accuracy gains over Naive Bayes. In Hier1 with 40 and with 160 features, for example, KDB-2 provides an accuracy improvement which is statistically significant ($P < 0.10$). In Hier3 with 80 features, the gains are even more dramatic, with KDB-1 providing an 80% reduction in error (with significance $P < 0.01$).

By contrast, the flat approach seems much less capable of taking advantage of the richer model space. In few cases did we observe improvements in accuracy with the KDB classifiers, and in no case was the improvement statistically significant. As mentioned previously, we conjecture that the reason for this shortcoming arises from the fact that the dependency models for the different classes are quite different. (This phenomenon is a form of *context-specific independence* [54, 14].)

The flat classifier is required to capture, within a single model, a complex dependency structure resulting from aggregating all of these disparate dependency structures. In this case, the dependencies are either "washed out" by the noise, or are so complex that it is impossible to capture them within the restricted dependency structure that we consider. Even if we built a series of binary classifiers, one for each individual topic (as we did with the Reuters-21578 data in Chapter 8, where each classifier simply predicts whether or not a document should be classified into a given topic), we would still need to build a complex dependency structure to capture the factors that seperated one class from *all* the others. Rather, we could simply build

Topic	10 most discriminating words
Top level	dollar, dealer, tonnes, agriculture, oil, grain, wheat, corn, gas, usda
Grain	london, taiwan, wheat, gulf, maize, k, corn, eep, enhancement, winter
Money Effects	dollar, japan, yen, money, england, repurchase, k, stg, shortage, system
Crude Oil	production, ship, gas, natural, iran, cubic, barrel, iranian, attack, tanker

Table 9.3: The 10 most discriminating words in one fold of the hierarchical method for the Hier1 dataset.

a set of binary classifiers in the context of the hierarchy (where at each topic node, a set of binary classifiers would simply predict whether or not a document should be filtered down to the various child topics of the node) and thereby still get the benefits of the context-sensitivity by making direct use of the topic hierarchy.

Thus, we see that the key to the success of our approach is the combination of three techniques: hierarchical classification based on the structured topic hierarchy, aggressive feature selection at each node of the hierarchy, and the use of richer dependency models. The use of the hierarchy serves to focus the attention of each classifier on distributions with more uniform characteristics (i.e., more semantically related topics), allowing us to target both the selected features and the dependency model to the *local* classification task.

To illustrate this phenomenon, Table 9.3 shows the set of 10 features (words) found to be most dicriminating at each level of the hierarchy learned during the run of one fold on the Hier1 dataset. At the top level of the hierarchy, we see a selection of high-level terms from the various major topics. Some of these are no longer indicative at the lower levels. For example, while the terms "agriculture" and "usda" are useful for identifying documents in the Grain topic, they are not useful for distinguishing among its subtopics. Rather, we see more specific words (such as "corn", "maize", and "wheat") that help distinguish between the two subtopics (Corn and Wheat) of the Grain topic. Similarly, the the Money Effects topic contains terms that help

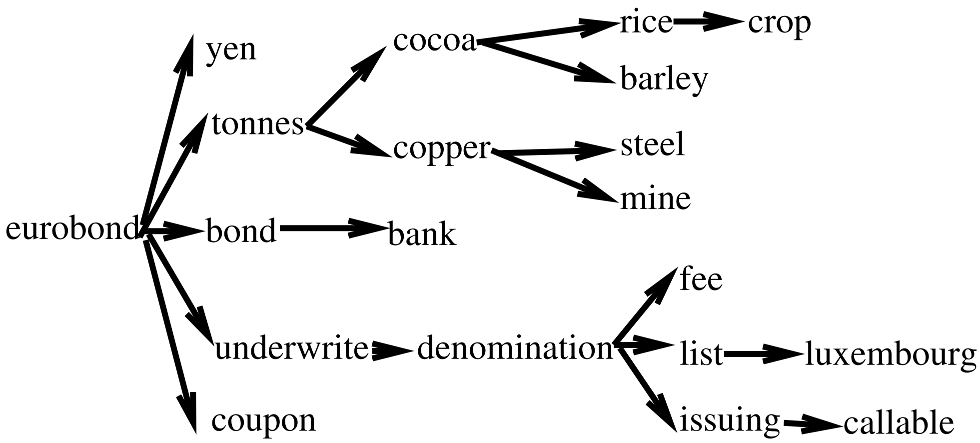

Figure 9.4: Dependencies between individual words found by KDB-1 among the 20 features selected in the top level of the Hier3 hierarchy.

distinguish documents about the Dollar (many of which relate to "japan" and the "yen") from articles that relate to Interest Rates. Finally, the feature selection for the Crude Oil topic autonomously homed in on all of the terms appearing in the names of its various subtopics ("natural", "gas", and "ship"). We also note that the features selected in the different cross-validation folds contain many of the same words at each node, thereby indicating that the method is generally robust in terms of selecting meaningful features across different partitions of the same dataset.

The same localization phenomenon allows the dependency model to be tailored to the relevant distribution. To illustrate this point, we examined the actual dependency structure between words constructed (automatically) by the KDB algorithm in the top level node in one randomly selected fold of the Hier3 dataset (where we obtained the biggest gains from the use of richer dependency models). Recall that the task at that node was to distinguish the topics *food commodities*, *metals*, and *money*. Figure 9.4 shows these dependencies (omitting the universal dependence on the class variable). We see that the word dependencies form small clusters, reflecting the correlations between the words in the domain. The cluster associated with the word "tonnes" is most intriguing, as the word is used for both metals and food commodities. The model constructs two subclusters connected to "tonnes", one related to metals and the other to food commodities. This dependency helps the classifier interpret the word "tonnes" as appropriate to the context.

Our approach allows us to utilize the synergy between these tools, resulting in significantly improved classification accuracy. To see this, compare the best hierarchical classifier versus the best flat classifier for each number of features (marked in boldface in Tables 9.1 and 9.2). In Hier1 and Hier2, we typically see small improvements, but not significant ones. We note that the hierarchical approach almost never detracts from accuracy, even though the hierarchical method requires the data to be fragmented at lower levels in the hierarchy. To examine the robustness of the hierarchical approach to fragmentation, we included the data-impoverished Hier2 dataset. As we see, even in this case, the hierarchical method never performed significantly worse than the flat method. In fact, it actually achieved the best results on this dataset (using KDB-2 with 40 features per node) although the high variance precluded any conclusion as to the significance of this improvement.

It is in the Hier3 dataset, which provided the largest hierarchical structure and thus the most classification information to leverage, that we see the true power of our approach. In comparing cases with equivalent numbers of features per classifier, we see that the hierarchical method significantly outperforms ($P < 0.05$) the flat classification scheme in *every* such comparison! We also compare the cases where an equivalent number of total features is used. The hierarchical method using 20 features per internal KDB classifer performs equivalently to the flat method using 80 features, but is far faster to train. Still more compelling is the fact the hierarchical method utilizing 80 features per classifier significantly outperforms ($P < 0.01$) the flat method using 320 features. Finally, we note that the hierarchical method (using KDB-1 and 80 features) achieves the highest overall classification accuracy on the Hier3 dataset, significantly outperforming *every* run of the flat method regardless of classifier or number of features used.

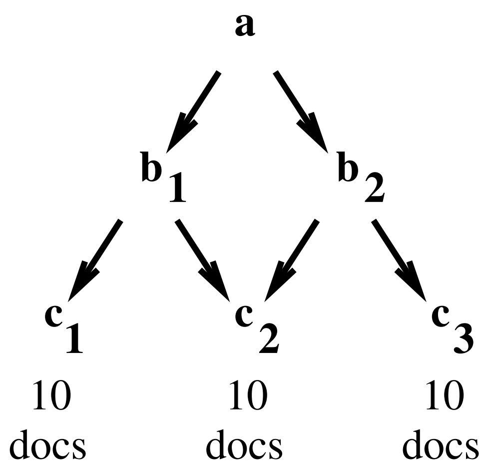

Figure 9.5: A sample directed acyclic graph of topics.

9.4 Extensions to Directed Acyclic Graphs

Thus far, the hierarchies we have considered have been tree structured. Some topic hierarchies, however, may not adhere to a strict tree, but rather may be represented by a directed acyclic graph (DAG). In such cases, it is still possible to apply our hierarchical classification scheme, but a bit more care must be taken so as not to count instances in the hierarchy multiple times. To illustrate this point, consider the sample hierarchy of topics (organized in a DAG) in Figure 9.5. Here, we use the variable B to refer to the classication in the first tier of the hierarchy and the variable C to refer to the final classification into the leaves of the hierarchy.

For the sake of simplicity, let us assume that a given instance vector $\boldsymbol{x}$ is equally likely to appear in any of the leaf classes c_1, c_2, or c_3. That is, $\mathrm{P}(\boldsymbol{X} = \boldsymbol{x} \mid C = c_1) = \mathrm{P}(\boldsymbol{X} = \boldsymbol{x} \mid C = c_2) = \mathrm{P}(\boldsymbol{X} = \boldsymbol{x} \mid C = c_3)$. Moreover, since each leaf class contains an equal number of instances, it ideally should be equally probable for $\boldsymbol{x}$ to be classified into each of the leaf classes. However, if we take too simple of an approach to using our hierarchical classification scheme within a DAG, and just classify the instance down each level of hierarchy into the class that has maximal probability, we run into the possibility of counting some instances multiple times. For example, consider the case of a document $\boldsymbol{x}$ being classified into class c_2. We would begin our hierachical classification of $\boldsymbol{x}$ by first either classifying it into b_1 or b_2, and then trying to classify into one of the final leaf classes. Since it is possible to reach class c_2 along a path through either b_1 or b_2, we must consider both possibilities in computing a final probability. Formally, this yields:

$$\begin{aligned} \mathrm{P}(C = c_2 \mid \boldsymbol{X} = \boldsymbol{x}) &= \mathrm{P}(C = c_2 \mid B = b_1, \boldsymbol{X} = \boldsymbol{x}) \cdot \mathrm{P}(B = b_1 \mid \boldsymbol{X} = \boldsymbol{x}) + \\ & \quad \mathrm{P}(C = c_2 \mid B = b_2, \boldsymbol{X} = \boldsymbol{x}) \cdot \mathrm{P}(B = b_2 \mid \boldsymbol{X} = \boldsymbol{x}). \end{aligned} \tag{9.2}$$

Since we assumed that the instance $\boldsymbol{x}$ actually provides no information with regard to the class label, we can compute the classification probability simply based on the class marginals (i.e., the number of training documents in each class). At the first level of the hierarchy, b_1 and b_2 each contain 20 documents in their descendents (i.e., the number instances classified into a leaf class that is reachable by following downward arcs from a node). Thus, to compute $\mathrm{P}(B)$, for example, we simply divide the number of instances in the descendents of each class B by the total number of documents in the descendents of all classes at that node. Such a computation yields

$$\mathrm{P}(B = b_1 \mid \boldsymbol{X} = \boldsymbol{x}) = \mathrm{P}(B = b_2 \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{20}{40} = \frac{1}{2}\ . \tag{9.3}$$

Similarly, the probabilities needed to perform classification at the second level of the hierarchy are given by:

$$\mathrm{P}(C = c_2 \mid B = b_1, \boldsymbol{X} = \boldsymbol{x}) = \frac{10}{20} = \frac{1}{2}\ , \tag{9.4}$$

and

$$\mathrm{P}(C = c_2 \mid B = b_2, \boldsymbol{X} = \boldsymbol{x}) = \frac{10}{20} = \frac{1}{2}\ . \tag{9.5}$$

Now, the problem of double counting becomes clear. By substituting the values from Eqs. 9.3, 9.4, and 9.5 into Eq. 9.2, we obtain:

$$\mathrm{P}(C = c_2 \mid \boldsymbol{x} = \boldsymbol{x}) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}\ . \tag{9.6}$$

If the instance $\boldsymbol{x}$ truly provided no information about the class, we would a priori

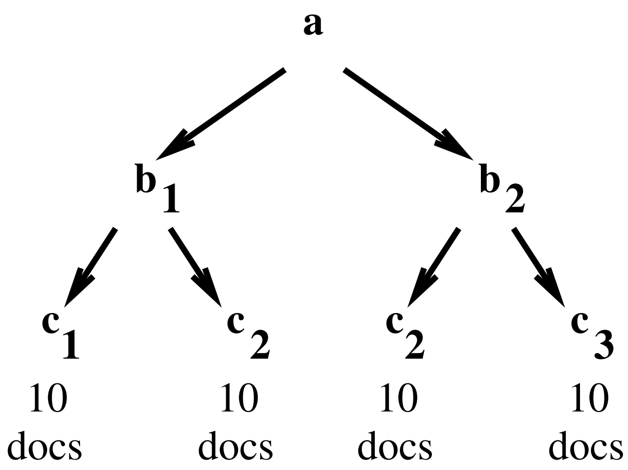

Figure 9.6: Treating the DAG in Figure 9.5 as a tree (without weighting) double counts some topics.

expect that each class c_1, c_2, and c_3 would each have probability $\frac{1}{3}$, since each class contains the same number of instances. However, in the calculations above, the instances in c_2 are being counted once for *each* path leading to this node. As a result, class c_2 has a disproportionately high probability, as evidenced in Eq. 9.6. In essence, our greedy classification approach has made it possible to double count the instances in c_2 since it is possible to reach this class via two different paths in the hierarchy. This situation would correspond to treating the DAG in Figure 9.5 as the tree structured hierarchy shown in Figure 9.6.

In order to solve this problem, we can divide the weight of a child node in the DAG among all its parents, with these weights summing to 1. For example, in the hierarchy shown in Figure 9.5, the instances in node c_2 could be equally weighted (with weight $\frac{1}{2}$) between its two parents b_1 and b_2. This weighting simply implies that each instance labeled as c_2 is counted as half an instance for all probabilities that need to be estimated (e.g., during both feature selection and classification) using the descendents of b_1, and similarly for the downward closure of b_2. Such a weighting would correspond to the hierarchy shown in Figure 9.7.

With this weighting scheme, we can now recompute the calculations in our previous example, showing that we do indeed obtain numerical results that are consistent

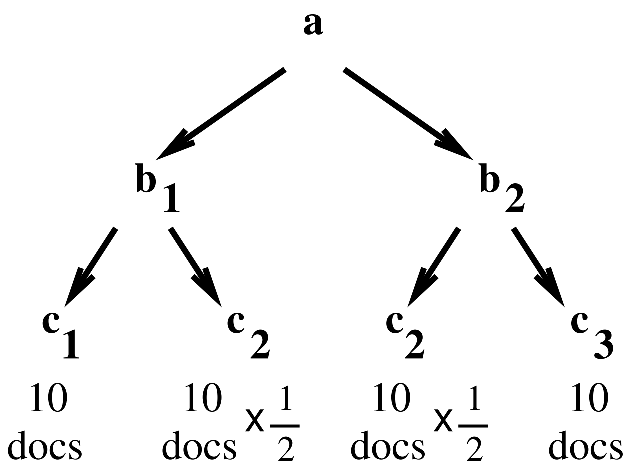

Figure 9.7: Weighting instances allows the DAG in Figure 9.5 to be treated as a tree, while not double counting instances.

with our intuitions about the a priori class probabilities. In the weighted scheme, we have

$$\mathrm{P}(C = c_2 \mid B = b_1, \boldsymbol{X} = \boldsymbol{x}) = \frac{5}{15} = \frac{1}{3}\ , \tag{9.7}$$

and

$$\mathrm{P}(C = c_2 \mid B = b_2, \boldsymbol{X} = \boldsymbol{x}) = \frac{5}{15} = \frac{1}{3}\ . \tag{9.8}$$

Note the the probabilities for b_1 and b_2 remain $\frac{1}{2}$, but are derived slightly differently.

$$\mathrm{P}(B = b_1 \mid \boldsymbol{X} = \boldsymbol{x}) = \mathrm{P}(B = b_2 \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{15}{30} = \frac{1}{2}\ . \tag{9.9}$$

Using these updated probabilities, we obtain the final result:

$$\mathrm{P}(C = c_2 \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{1}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{1}{2} = \frac{1}{3}\ , \tag{9.10}$$

which matches our intuitive expectations (and actual distribution of instances among the leaf classes).

More generally, in order to extend our work to DAGs we need to consider two factors. First, nodes in the DAG with multiple parents should be probabilistically weighted among their parents (as noted above), so that the instances contained in their descendents are not counted multiple times during probability computations. In this way, we can guarantee that each instance in the data is, in fact, only counted once regardless of the structure of the topic hierarchy. Second, the probability of predicting a given class c should be determined as the sum over all paths in the DAG leading to that class node. Formally, this can be expressed as

$$\mathrm{P}(C = c \mid \boldsymbol{X}) = \sum_{Y \in \{\text{paths to } c\}} \mathrm{P}(C = c, Y \mid \boldsymbol{X}) \ , \tag{9.11}$$

where Y is a vector of nodes on the path from the root of the DAG to class node c.

While this sum may appear to become computationally unmanageable due to the possible exponential blow-up in the number of paths to a given node in a large hierarchy, it is important to keep two considerations in mind. First, the connectivity in topic hierarchies is often quite sparse, thus strongly controlling the number of possible paths to a given node. Second, even in cases where the number of paths to a given node may be very large, it is still possible to efficiently compute the sum in Eq. 9.11 using dynamic programming.

By using a dynamic programming solution, we can decompose Eq. 9.11 into sums over local path segments, thereby eliminating the exponential growth in the total number of terms (i.e., complete paths) that would normally have to be computed in the sum. This decomposition is achieved by simply storing the probability of an instance reaching each node in the graph, rather than maintaining a separate probability for each possible path that the instance may take through the graph. As a result, for each node in the topic hierarchy, we only need to sum over the probabilities of getting to it from its immediate parents (weighted by the probability of an instance reaching each of the immediate parent nodes). Thus, even if there are exponentially many distinct paths leading to some node in a topic graph, we can still compute the probability of reaching that node in time that is polynomial in the total number of nodes in the graph.

In future work we would like to more empirically explore the issue of hierarchical classification into topic DAGs. However, the current implementation of SONIA only requires classification into strict trees of topics, and hence we do not pursue classification into DAGs more fully here. Still, while incorporating topic DAGs into SONIA may be well understood from a computational point of view, it is not a trivial task when considering an appropriate user interface for developing such hierarchies. This is yet another reason why we do not address the more empirical issues here.

9.5 Conclusions

The construction of a system such as SONIA that can help hierarchically organize large amounts of text-based documents calls for algorithms that hierarchically categorize new documents as they come in (e.g., in response to users' queries). We described an approach which utilizes the existing rich hierarchical topic structure in order to facilitate this process. Rather than building a single massive classifier, our approach generates a hierarchy of classifiers, utilizing feature selection to tailor the feature set of each classifier to its *localized* task. As we have shown, the resulting reduction in the size of the classifier allows us to obtain significantly higher accuracy, a reduction due both to increased robustness and to our ability to use richer (and more complex) classifiers.

In other domains, such as medicine, the argument has also been extended that making use of structure in the problem can help to reduce the complexity of a diagnosis task. For example, in the Pathfinder system [74, 75], it was observed that by constructing a single model for several related diseases, it was possible to build more robust and specialized diagnosis models for different groupings of diseases. While, this system was not focused on a classification task per se, the results obtained in that work lend further credence to the benefits of making explicit use of class relationships, such as in our hierarchical scheme.

Our work also opens the door to using more expressive (and computationally more expensive) classifiers at the nodes of the hierarchy. In this way we hope to be able to obtain not only better classification results, but also be able to handle text collections with a wider variety of statistical characteristics.

Several research issues specific to hierarchical classification still remain. In particular, we have already mentioned the problem of recovering from classification errors early in the hierarchy. Moreover, recent work by McCallum *et al* [116] on improved methods for estimating probabilities for classification in text hierarchies are directly applicable in our work and may even further improve our results. We would also like to investigate the problem of discovering new classes in the hierarchy, when we have multiple documents that do not "fit in" nicely. We believe that incorporating some of the clustering work from Chapter 6 may help address this issue in future work.

Finally, in addressing the issue of applying our method to large text hierarchies existing on the Web, we have already conducted some preliminary work on applying this method to sub-hierarchies of topics extracted from the *Yahoo!* Web directory. These results are still inconclusive (as Web data has a tendency to be extremely varied, as well as incorporating other media aside from text), but we are encouraged by some of our initial results and are seeking ways to improve them. More recently, Mladenic [118] has directly followed up on this work, making use of our hierarchical classification scheme also for classifying Web pages into the *Yahoo!* directory. Her positive results on this task lend further support to the applicability and scalability of this method on large text datasets.

Part IV

Putting It All Together

Chapter 10

SONIA – A Complete System

10.1 Introduction

Having described the major technical components that comprise SONIA in the previous chapters, we can now see how all these pieces fit together to form a complete system. To return to our original motivation for building SONIA, recall that the enormous amount of information available on the World Wide Web and other networked information sources, such as Digital Libraries, has created an urgently pressing need to provide users with tools to navigate these information spaces. The initial attempts at addressing this problem have led to the development of a number of information finding tools such as Web-based search engines (e.g., *Alta Vista*) and hierarchical directory services (e.g., *Yahoo!*). However, as we explained previously (in Chapter 1), such methods for information access are quickly being rendered inadequate due to the tremendous growth in the number of documents available. Also worth noting is the fact that networked information can often come from a number of heterogeneous sources (i.e., the World Wide Web, different Digital Libraries, proprietary databases, etc.), whereas many existing information finding tools are only implemented to work with one information source.

We seek to address these problems with SONIA, a system for *topical* information space navigation that combines both the query-based and taxonomic approaches. SONIA employs the machine learning techniques described earlier (i.e., feature selection, clustering, and hierarchical classification) to create dynamic document categorizations based on the full-text of articles that are retrieved in response to users' queries. In this way, users can explicitly specify their information needs as queries while also having the ability to browse the results of their queries at a topical, rather than document, level. Moreover, the hierarchical organization scheme imposed on a set of documents may be used to classify new documents that become available to the system (e.g., documents returned in response to follow-up queries by the user).

The ability to automatically create meaningful document groupings relies on the validity of the Cluster Hypothesis. Recall that this hypothesis states that "closely associated documents tend to be relevant to the same requests" [166]. As we described in Chapter 4, a good deal of previous work [167, 127, 70, 71, 2] has provided significant support for the cluster hypothesis, as have our more recent results reported in Chapter 6. Thus, we have strong reason to believe that clustering may be used as an effective tool to help organize documents.

While our system embodies some similar elements to those in previous work on document clustering, it uses entirely different, and in many cases improved, technologies to realize this functionality. More importantly, however, SONIA provides significant new extensions in terms of technical functionality and broader applicability. Operating in the dynamic context of networked information, SONIA makes use of a number of methods for relevant feature extraction from documents through a multi-tiered feature selection process that is customized to each user query.

The most significant extension of SONIA beyond existing systems, however, is the ability to save various document clusterings (i.e., topical partitionings) as hierarchical classification schemes that can be used to automatically categorize the results of subsequent, but related, queries (using the hierarchical classification method presented in Chapter 9). This combination of clustering and classification allows users to not only navigate a given document collection more easily, but enables them to quickly construct and maintain their own organizational structures for the vast quantities of information available to them. In this way, we hope to elevate user interaction with large information sources (e.g., the Web, Digital Libraries, etc.) beyond simple one-shot queries and move to addressing users' more persistent information needs.

We show that the basic ideas implemented in SONIA are also quite applicable to personal information management, such as helping a user organize the file system on his or her personal computer desktop. In this context, SONIA can be very effective at helping automatically generate and maintain portions of a user's hierarchical file directory structure.

Finally, since our system exists as part of a general architecture within the Stanford Digital Libraries Testbed [158], it has the ability to simultaneously retrieve information from a number of heterogeneous sources, thereby making our system maximally flexible. SONIA was also designed with efficiency in mind, thereby facilitating real-time user interactivity even when accessing diverse, distributed document collections.

In the remainder of this chapter we present the technical details of SONIA. In Section 10.2, we describe the architecture in which SONIA is embedded and how it interacts with a variety of heterogeneous information sources. There, we also give a brief description of the Stanford Digital Libraries InfoBus Architecture, showing how SONIA is situated within a larger distributed systems context. Section 10.3 provides a more detailed account of how the various machine learning methods employed in SONIA are incorporated together in the system. Then, in Section 10.4, we show anecdotal examples of the system in use, discussing its efficacy in information browsing and classification. Finally, Section 10.5 gives a summary of this work and its future directions.

10.2 SONIA on the InfoBus

The focus of the Stanford Digital Libraries project is on providing interoperability among heterogeneous, distributed information sources, services and interfaces. To this end, the InfoBus architecture [10] shown in Figure 10.1 has been developed. In brief, the InfoBus is comprised of network proxies that encapsulate the protocols used by disparate interfaces, information sources, and information services. These proxies allow for communication among the different entities connected to the InfoBus by translating their communications into a common language.

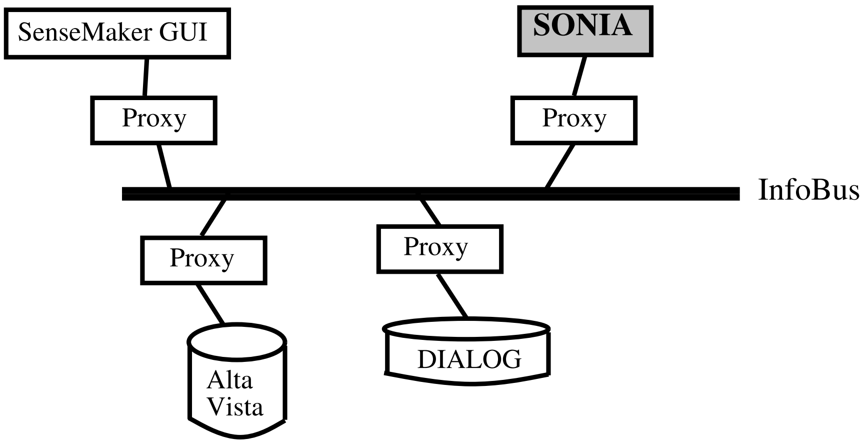

Figure 10.1: The InfoBus architecture.

The InfoBus thus provides a mechanism whereby a number of different information sources may be uniformly accessed. Furthermore, as new information sources are incorporated into the InfoBus, their contents may be accessed with minimal re-engineering of existing systems. In this way, systems that make use of the InfoBus protocols are easily adapted to account for the emergence of diverse new information sources. Also, the InfoBus provides a number of built-in services, such as intellectual property rights management and automatic payment for information, among others. These additional services are beyond the scope of the functionality necessary for describing SONIA's relationship to the InfoBus, and hence we do not describe them further here.

SONIA exists within this architecture as an information service with a number of capabilities. First, it allows for the clustering of collections of documents to help extract subtopics that may be present. Furthermore, SONIA also allows for such document groupings to be stored as persistent hierarchical categorization schemes. Each such hierarchy is simply a multi-level partitioning of documents into a number of semantically meaningful groups. These hierarchies may then be updated by automatically classifying additional documents into them. In this way, new query results can be integrated into an existing topic hierarchy derived from previous query results. As a result, a user can build up a large collection of results spanning multiple related queries (possibly directed at different information sources) within the same organizational scheme. Moreover, SONIA allows a single user to save several distinct hierarchies to reflect each of his or her diverse information needs.

Previously, the functionality in SONIA was accessible through the Java-based *SenseMaker* interface [11]. SenseMaker allows users to simultaneously query multiple heterogeneous information sources and then organize the retrieved documents by matching titles, matching URLs (for Web documents), and the like, or it can utilize SONIA to cluster documents by their full-text content [142, 143]. However, since the the focus of the SenseMaker interface was not on the formation of topic hierarchies, we have more recently developed our own custom interface for SONIA. In this new interface (which we show examples of in Section 10.4), users can still query a variety of heterogeneous information sources, but can now form hierarchical organization schemes to capture the richer topical structures that exist in many document collections. The InfoBus is still used as the primary means of communicating with these disparate information sources. Thus, SONIA can effectively query virtually any information sources which are, or will be, connected to the InfoBus.

10.3 A Component View SONIA

It is simplest to view SONIA as a series of modules, each of which is responsible for a data transformation procedure. We begin by examining the initial querying and document processing stages in SONIA, which are represented in the block diagram in Figure 10.2. Here, the user begins by using SONIA to issue a query to various information sources linked to the InfoBus. A list of potentially relevant document identifiers (e.g, URLs for Web documents, ID numbers for DIALOG, etc.) are then returned to the system in response to the query. SONIA then employs a parallelized crawler to retrieve the full text of the documents referred to in this list of document identifiers.

At this point, it becomes necessary to represent these documents in a manner suitable for processing by subsequent machine learning algorithms (i.e., a vector representation). The processing stages which produce this vector representation very

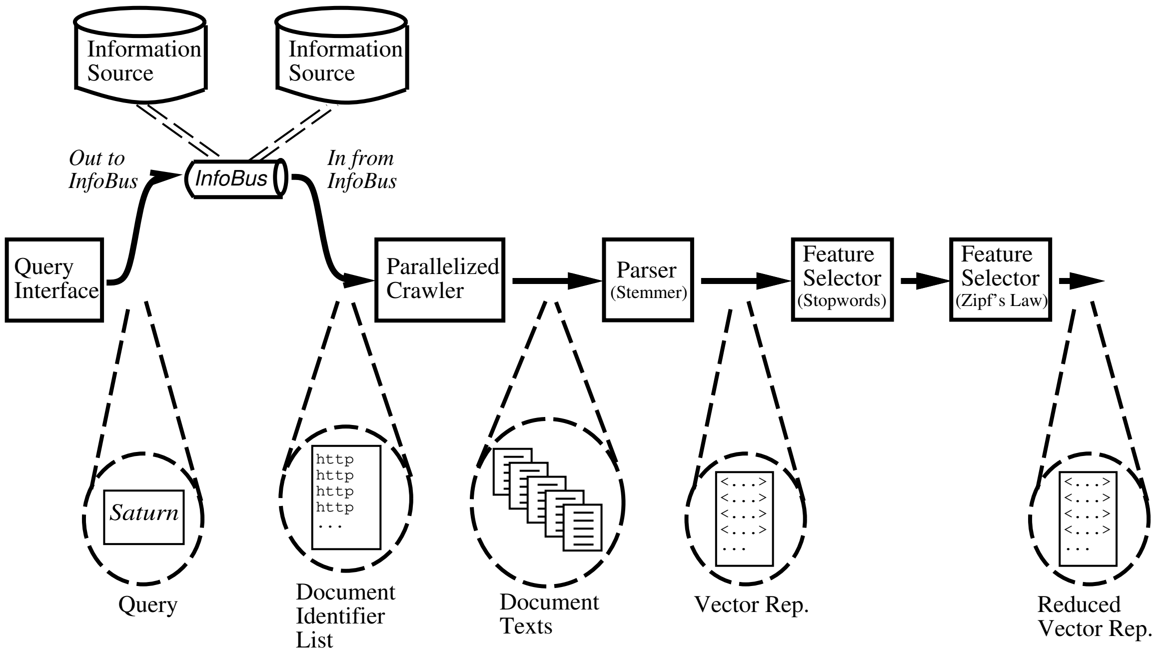

Figure 10.2: Initial querying and document processing stages in the SONIA system.

closely follow our treatment of document representation given in Chapter 2. Once a suitable representation has been obtained, the documents are then analyzed by different machine learning tools, depending on whether the user is choosing to organize the documents according to an existing hierarchical classification scheme or not. The machine learning components of SONIA are depicted in Figure 10.3. We explain the details of all of these processing stages, as implemented in SONIA, below.

10.3.1 Document retrieval and parsing

Since on-line information sources are rapidly changing, SONIA does not attempt to maintain its own (possibly outdated) inverted index of documents, but rather treats networked information as a massive digital library from which it can dynamically retrieve documents. As a result, SONIA only requires that it receive a list of document identifiers (and not the actual documents) from the information sources that it queries. This makes it possible to employ SONIA in conjunction with any of the existing well-known search engines on the Web.

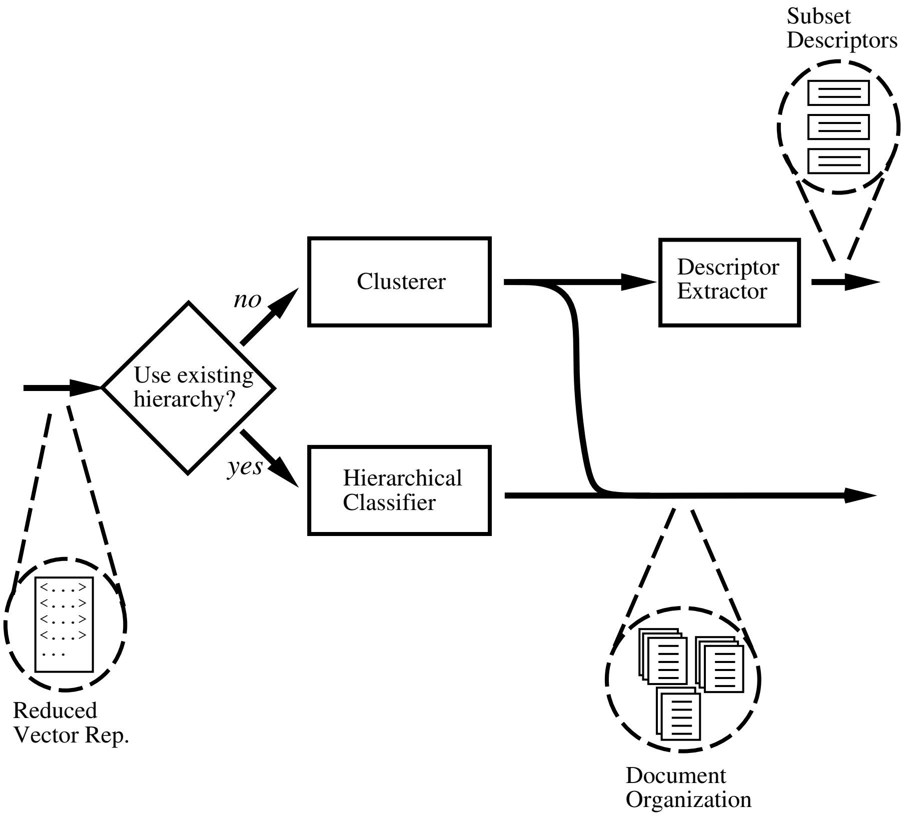

Figure 10.3: The machine learning components in the SONIA system.

A highly parallelized document retrieval module (sometimes called a network *crawler* or *spider*) is employed to retrieve the full text of the corresponding documents. This module does not present a timing bottleneck in real-time interaction as it is capable of robustly retrieving as many as 250 document texts in parallel, and utilizes a time-out condition (currently set at 30 seconds) to prevent needlessly long waits for documents. Moreover, as the infrastructure of the Internet continues to improve, and localized repositories become more prevalent, this document retrieval stage will become even less of a time issue.

The retrieved document texts are then parsed into a series of alphanumeric terms (i.e., words). Optionally, these terms may be stemmed to their root as SONIA's parser includes a standard word stemming scheme [128]. We note that we currently do not make use of such stemming in the examples of system usage provided in later sections. Empirically, we have not found stemming to create much of a difference in the results obtained with the system.

Each term then forms a dimension in a high-dimensional space in which the documents can now be represented as vectors. That is, the vector representing a document contains in the vector entry for each term, the count of how many times that term appeared in the document. Since we now have the term counts for each document, SONIA is capable of transforming the vector representation of documents to different term weighting schemes, such as a Boolean representation (as in Eq. 2.6). Such different representations are easily generated when needed by different modules within SONIA.

10.3.2 Initial feature selection

As we have mentioned on a number of occasions, the number of distinct terms in unrestricted text is very large, so feature selection is generally necessary. SONIA uses a multi-tier feature selection process, using both stop word elimination and statistical techniques to reduce the feature space drastically. The system currently incorporates two initial forms of feature selection, each of which operates on the vector-space representation of the documents. Initially, dimensions representing stop words are eliminated from the document vectors. These stop words are determined using a standard English stop word list of 570 words (for example, see Table 2.4), as well as a hand-crafted list of approximately 100 Web stop words (such as "html" and "url").

In the second tier of feature selection, a Zipf's Law analysis of term occurrence over the collection is used. This process eliminates terms that appear fewer than three or greater than 1000 times in the entire collection as not having adequate resolving power to differentiate sub-collections of documents. These threshold values implemented in SONIA were chosen heuristically since they appear to work quite well in practice across the range of collection sizes typically used with the system—a few hundred documents. We note that we have explored methods for tuning these thresholds to the characteristics of more varied collections in previous work [141]. However, in the current implementation of SONIA we have not observed much empirical difference through the use of these more sophisticated techniques, and thus do not employ them here to save processing time.

After these first two stages of feature selection, the system reaches a branching point (see Figure 10.3) depending on the user's choice to organize the current set of documents with respect to an existing hierarchy or not. If an existing topic hierarchy is not being employed, we are working in the context of *unsupervised* learning, and consequently allow the user to create a new document organization from scratch using the clustering method described in Chapter 6. If an existing hierarchy is being used, then we have a *supervised* learning problem, in which case we can employ the hierarchical classification scheme presented in Chapter 9. We describe the clustering and classification modules in more detail in Sections 10.3.3 and 10.3.5, respectively.

10.3.3 Clustering

We first consider the case where an existing hierarchy is not being employed and the user chooses to create a new organization scheme. Since we have found that the clustering techniques presented in Chapter 6 are quite effective without the use of any additional feature selection beyond that which we have already applied, SONIA currently does not make use of the unsupervised feature selection methods presents in Chapter 5. This not only gives us an important computational savings, but we believe that the use of additional feature selection at this point would have little influence on the efficacy of the clustering algorithm. Moreover, the clustering algorithms we employ are less computationally intensive than those used for classification. Hence the fact that we keep more features during clustering is not a serious computational hindrance.

In the clustering method implemented in SONIA, we use the document similarity measure based on the scaled term overlap between two documents, given in Eq. 6.19. To compute the probability of a word appearing in a document, we used the normalized geometric mean (NGM) estimate, defined in Eq. 6.10, as this estimate provided the best results in our previous comparative work. As in Chapter 6, we use the *group-average* measure (defined in Eq. 6.18) to determine the similarity of two *clusters* using a similarity function defined over two *documents*. Nevertheless, we note that any reasonable clustering method can be used in this module of SONIA. In fact, SONIA's modular architecture makes it easy to upgrade any individual component as advances are made in a particular technology.

In conjunction with this similarity measure, SONIA makes use of a two step clustering procedure. As explained in Chapter 6, this clustering procedure first uses hierarchical agglomerative clustering to form an initial set of seed clusters. These clusters are then further optimized using an iterative clustering technique (with a maximum of 10 iterations). We chose this two step approach since it produced the best results in our previous experiments. Moreover, this approach is deterministic, which is an important characteristic from the end-user standpoint. Our belief is that reproducibility of results is important to users, especially since they may find it disconcerting to see different results when applying a clustering operation to the same collection of documents.

Finally, note that the clustering methods we employ currently require that the user specify a priori the number of clusters into which a collection of documents should be grouped. SONIA provides an interface which easily allows users to direct the system to produce anywhere from two to 10 clusters. Moreover, since SONIA also allows users to *aggregate*, or undo a clustering of documents, it is simple for users to repeatedly try clustering a collection of documents into different numbers of subgroups and then select the one that best suits their needs.

10.3.4 Descriptor extraction

Whenever the user clusters a set of documents, he or she not only produces a partitioning of the documents, but also causes SONIA to automatically extract *descriptors* from the document subsets. These descriptor words provide a description of the subtopics found in the document collection, and are presented to the user as initial labels for each cluster.

We have informally compared a few methods for extracting these descriptors.

The first such method is a probabilistic *odds* scheme in which, for each cluster c_j, we compute the probabilistic odds $O_j(x_i)$ of a term x_i appearing in a document in c_j versus appearing in a document in any other sibling cluster c_k. More formally, we have

$$O_j(x_i) = \frac{\mathrm{P}(x_i \mid c_j)}{\sum_{c_k \neq c_j} \mathrm{P}(x_i \mid c_k)} \ . \tag{10.1}$$

We then select some number, κ, of terms with the highest O_j values as the descriptor for document subset c_j. Currently, we use $\kappa = 12$, as this value appears to achieves a good balance between brevity and descriptiveness. Alternatively, we have also considered a simple *centroid*-based approach for descriptor extraction. Here, we simply compute the Euclidean centroid of all documents assigned to each group c (using the term frequency vector representation for each document). As before, we simply take the κ terms corresponding to the dimensions with highest value in the centroid vector as the descriptor for that group.

In practice, we have found that the centroid-based approach appears to yield words that are much more indicative of the topic of a given document subset. It should be noted, however, that part of the success of the centroid-based approach relies on the efficacy of prior stop word elimination to prevent common meaningless words from appearing in the descriptor lists, since these words will be very common and hence have high frequency counts in all document subsets. In contrast, the problem with the odds based approach is that it seems to favor very rare (and hence not particularly descriptive) terms that may appear a few times in one document subset, but not in any of the others. As a result, these terms get a much higher *odds* score than more common terms that may appear even a few times in the other document subsets. Still, it may be of interest to select the terms to present to the user with the centroid-based method, but then order the terms according to the odds-based method. Pursuing this point in detail, however, is beyond the scope of our current work, and we do not currently make use of this idea in the system.

We also make use of the descriptor extraction module in SONIA to help *suggest query terms* pertaining to a particular document subset to the user. Here, the user simply selects any topic node (i.e., document subset) in a hierarchy and asks the system to supply terms which may be useful in future queries aimed at finding more documents related to the selected topic. We simply employ the same centroid-based descriptor extraction scheme, and list for the user the top 50 words (in order) in the resulting centroid vector. As will be seen in the example usage scenarios in Section 10.4, we have found this scheme to work quite well in practice.

10.3.5 Classification

Rather than using clustering, a user may be employing an existing topic hierarchy to classify an incoming stream of documents (for example, in response to a follow-up query by the user). In this case, SONIA directly applies the hierarchical classification scheme (i.e., the *pachinko machine*) described in the previous chapter. To this end, we use a combination of feature selection and Bayesian classification at each node in the user's existing topic hierarchy. Here, the documents in the existing hierarchy simply become the training data and a hierarchy of classifiers is built that can then be used to classify the new incoming documents. In generating the hierarchy of classifiers, we first apply the feature selection algorithm introduced in Chapter 7 to the documents at each node of the classification hierarchy. Note that we use no conditioning information (i.e., Markov blanket size = 0) during this feature selection process. This simplification allows us to obtain fast computational speed, without much degradation in classification performance (as evidenced by the results in Chapters 8 and 9). Currently, we select the 50 most informative features at each node, since the results from Chapter 7 show that 50 features are often sufficient for accurate classification. Moreover, using so few features gives us a big win in terms of computation time when subsequently inducing a classifier.

SONIA's architecture is general enough to allow any classification algorithm to be used as part of the system. We have chosen to focus on techniques based on k-dependence Bayesian classifiers presented in Chapter 8. More specifically, SONIA makes use of 1-dependence models, as these seem to provide the best tradeoff between expressivity and parameter robustness (i.e., bias and variance).

Once the hierarchy of classifiers is built, we simply classify the new documents

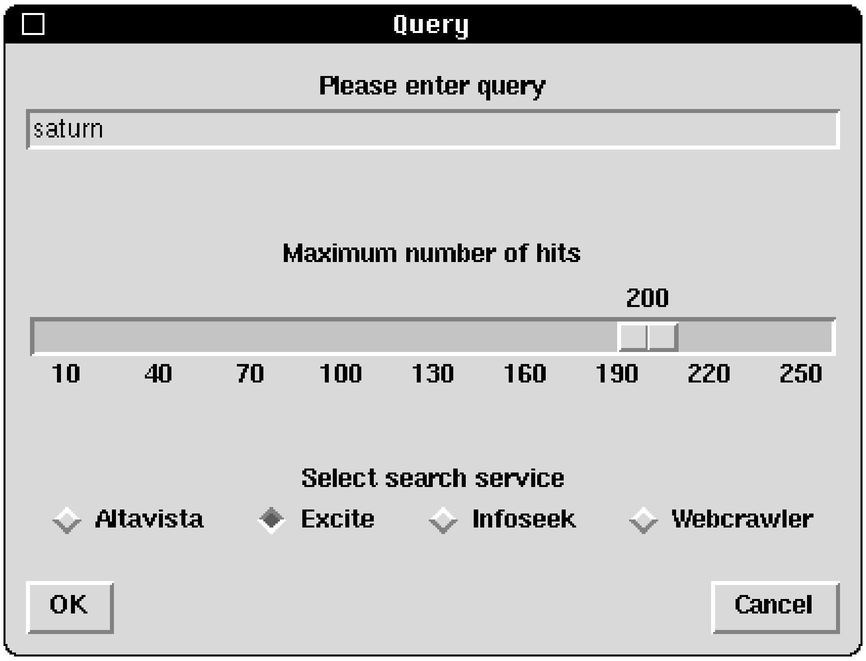

Figure 10.4: Using SONIA to issue the query "Saturn" to *Excite*.

in accordance with this resulting pachinko machine. After the new documents are classified, they are then displayed in the appropriate hierarchy nodes with asterisks appended to the start of their titles to readily differentiate them from the documents previously existing in the hierarchy. We show an example of this in the usage scenarios below. Thus, the topic hierarchy can be updated with minimal effort by the user.

10.4 Examples of System Usage

Having detailed the myriad components that comprise SONIA, we now give detailed examples of the complete system in action. Since it is difficult to provide an objective measure by which to evaluate such a system as a whole, we stress the salient aspects of each of the examples presented here, so as to highlight both the strengths and weaknesses of the system.

10.4.1 Usage Scenario One

In the first usage scenario, let us consider the situation in which a user is interested in finding out more information about the ringed planet Saturn and its recently discovered new moons.[1] The user begins by using SONIA to issue the query "Saturn" to the *Excite* Web search service, asking for the top 200 matching URLs. The query dialog box in which this request is issued is shown in Figure 10.4. Note that the user has a uniform interface regardless of which information sources are being queried.

SONIA sends this query via the InfoBus to *Excite* and receives the top 200 matching URLs back from the search engine. At this point, the parallel crawler module of SONIA is invoked to simultaneously retrieve all 200 Web pages pointed to by these URLs. Of the 200 original URLs, the crawler is able to retrieve 150 valid documents, since many links on the Web are outdated and point to non-existent documents, or the servers on which some of these documents reside may not be currently operational. These documents are then parsed into a vector representation, which initially has approximately 4000 features. The application of both Zipf's Law-based feature selection and stop word elimination reduces these vectors to 1872 features. The entire process of document retrieval, parsing and initial feature selection takes approximately 1.5 minutes of wall clock time on a heavily loaded Sun SPARC ULTRA 2.

The user then chooses to cluster this initial group of documents into four clusters in order to see what sorts of topics are contained in this collection. Applying clustering to the entire collection and then extracting descriptors from each resulting cluster takes roughly 1 minute. While we believe that the time taken to cluster is quite reasonable for an interactive system, we note that it would be possible to speed-up this process in many ways if desired (e.g., using an approximation to the similarity measure that is faster to compute, performing fewer clustering iterations, etc.), but such speed-ups may have a negative impact on the quality of the resulting clusters. However, it may be possible to tune our system for different user contexts, as appropriate.

An overview of the results of this clustering are given in Table 10.1, which includes

[1]This example was run in the summer of 1998. Since the Web is very dynamic, it is likely that running this same example at a later date will produce different results than those reported here. As with most things in life, *your mileage may vary.*

Extracted Descriptors	Sample Document Titles	Plausible Topics	No. Docs
saturn sega games sale fighter game virtua world nhl www ii $5	Sega Online: Strategy Guides Sega Saturn with 6 games for sale Sega Saturn Links	Sega Saturn Video Game	23
saturn car club price saturn's market higher san 000 service money diego	Saturn: A Case Study of How to Grow Saturn Falling On Hard Times Saturn of Honolulu	Saturn Car Businesses	19
saturn talk car www sc2 fl kind mail 4a4 home dark sl2	Saturn, Let's Talk Saturn New Car Sales Saturn is different! New cars and used	Saturn Car Talk and Info	70
saturn saturn's rings moons ring jupiter moon planet system hydrogen interior earth	Saturn's Small Moons The 1995-6 Saturn Ring Plane Crossings Science Tip - Saturn	Planet Saturn	38

Table 10.1: Initial clustering results on documents matching the query "Saturn".

for each cluster the descriptors automatically generated for that cluster, a sample of the titles of documents contained in the cluster, a human-generated description of the likely "plausible topic" for the cluster, and the number of documents in the cluster. Figure 10.5 shows the SONIA interface after this clustering, reflecting the first level of the topic hierarchy being generated by the user. Figure 10.6 shows this hierarchy after the user has renamed each cluster with its plausible topic.

The results in Table 10.1 reveal that SONIA is quite capable of discovering meaningful subtopics within the given collection, readily distinguishing those documents about the planet Saturn with those about the car company, as well as the Sega Saturn video game. Moreover, these results also highlight how query results may contain many documents which are about topics entirely different than what the user is truly interested in. By using clustering to quickly identify and sort out these non-relevant documents, the user may be able to better home in on just those documents relevant to him or her. This feature is especially important when we note that some of the web pages of Saturn car enthusiasts have such vague titles as "Saturn Page" and "Saturn" that could be misconstrued as pages about the planet if only titles were available (as is the case with simple Web searches that provide no categorization mechanism). Indeed, a detailed analysis of this document collection reveals that only 38 of 150

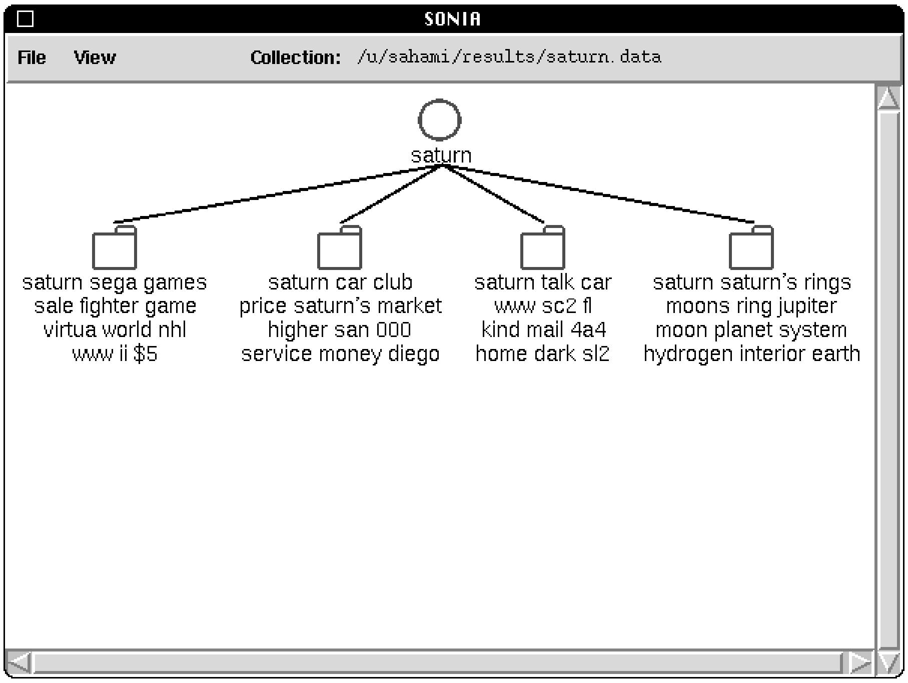

Figure 10.5: SONIA display after initial clustering of "Saturn" collection.

documents are about the planet Saturn, and all of then are correctly grouped together into one cluster. Similarly, the collection contains 23 documents about the Sega Saturn video game, which are all placed in a single cluster. The remaining two clusters contain documents exclusively about the Saturn car company. Thus, in this example, it appears that SONIA's clustering algorithm is very effective at keeping each subgroup of documents relatively coherent.

At this point, the user wishes to find more structure in the subcollection of document specific to the planet Saturn. Consequently, SONIA is used to cluster this subgroup into three clusters. A brief overview of these clusters is given in Table 10.2. A closer examination of the contents of each cluster reveals that one contains documents on general information about the planet Saturn itself, another contains documents primarily about Saturn's rings and moons, and the final cluster contains documents pertaining to the observation of Saturn. We do note, however, that the descriptors automatically extracted from each cluster tend to have several overlapping terms. Thus, using these descriptors alone may be insufficient for the user to properly discern the

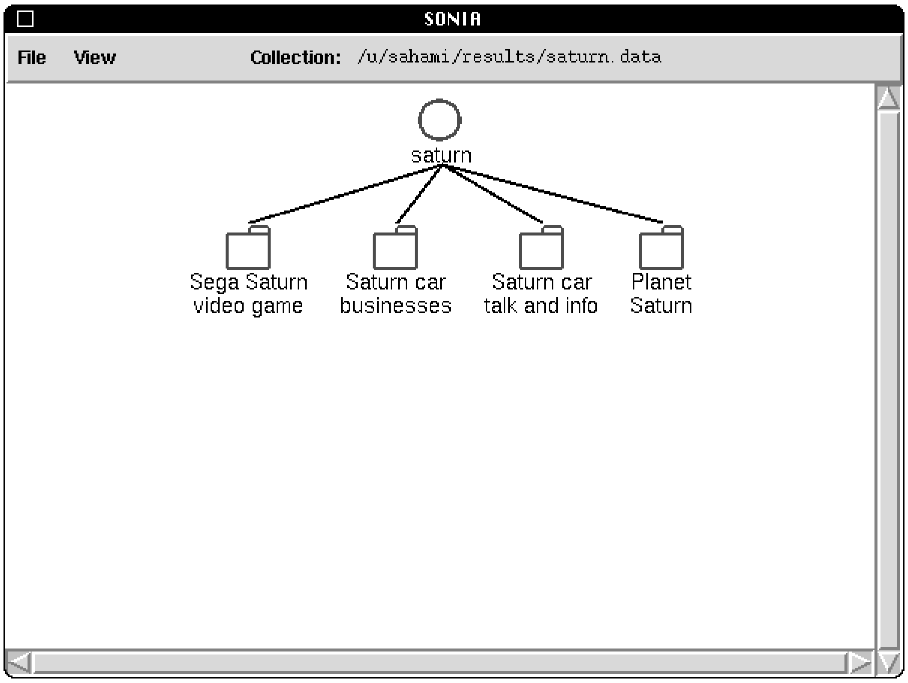

Figure 10.6: SONIA display after naming of the document clusters.

true topic of each subcollection.

The reduced efficacy of our descriptor extraction scheme lower in the hierarchy is mostly attributable to the fact that topics lower in the hierarchy tend to be much more related to their sibling topics. The documents in these related fine-grain topics use many of the same words. Thus, the documents generally become less distant from one another. Hearst also discusses this point in her work on information access user interfaces that make use of clustering [69], where she explains that the focused results of complex queries may be difficult for clustering systems to meaningfully differentiate. Still, by reading just a few of the document titles in each subcollection, the contents of each subgroup becomes much more clear.

A close manual examination of this clustering shows that only two documents are questionably placed in a cluster which may not best match their topical theme. These documents both have a fair deal of information about the planet Saturn itself, but also contain a good deal of information on Saturn's rings and virtually all of its moons. These documents are both clustered into the "Moons and Rings", which seems

Extracted Descriptors	Sample Document Titles	Plausible Topics	No. Docs
saturn hydrogen planet interior saturn's jupiter ice layer rings composition system core	Saturn Facts Composition of Saturn's Interior Saturn - What We Know	General information	17
saturn saturn's moons rings ring moon plane jupiter system voyager image planet	7 (Saturn) Moons, compare Saturn's Small Moons The 1995-6 Saturn Ring Plane Crossings	Moons and Rings	16
saturn jupiter moon venus mars mercury day rings side earth picture visible	Best times to observe Hourly Cycle of Solar System Objects APOD: July 5 - Night Side of Saturn	Observation	5

Table 10.2: Results of clustering the "Planet Saturn" subcollection.

quite acceptable. However, they could have been equally reasonably grouped into the documents containing general Saturn information. While their current grouping is very reasonable, this example does show that allowing documents to belong to *multiple* clusters would be a desirable property. In previous work [141] we have taken some initial steps in this direction, but a good solution to this problem still remains an open question.

Still, if the user believes that some documents are improperly categorized by the system, SONIA's interactive interface allows the user to move such documents to other folders in the hierarchy if they choose. Thus, the clustering need not be 100% accurate in order to still be effective at partitioning the document collection into meaningful subgroups.

Returning to the usage scenario, the user now decides to rename the subcollections about the planet Saturn for easier identifiability. Moreover, the user also restructures part of the hierarchy dealing with Saturn cars to create one subtree which encompasses both of the document groups related to this topic. This final hierarchy, as displayed in SONIA, is shown in Figure 10.7.

Focusing on the subtree dealing with the planet, Figure 10.8 shows a view of the interface with two of the document folders open. This figure displays how users are able to browse document titles (and accompanying URLs) easily within the SONIA interface. Thus, when automatically extracted descriptors may be inadequate to

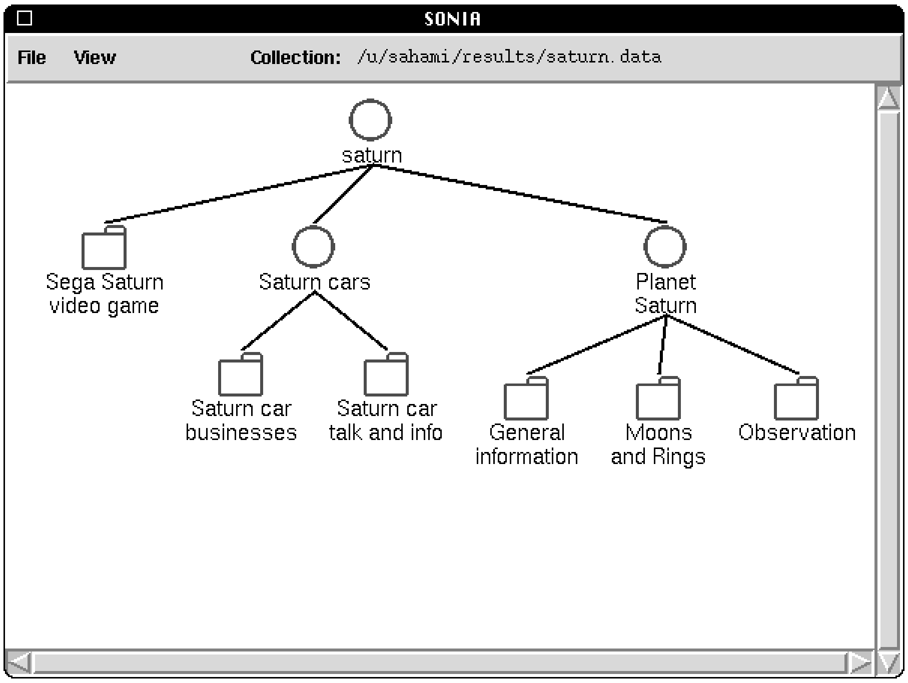

Figure 10.7: SONIA display showing the hierarchical organization produced by the user.

characterize the topic of a group of documents, it is simple enough to peruse the document titles within SONIA.

Nevertheless, a user may wish to browse the actual Web pages in a given group of documents, as opposed to just their titles. To this end, SONIA is capable of dynamically generating Web pages which contain links to the actual documents contained in each node of the hierarchy. Moreover, a browser is launched in parallel with SONIA to display such Web pages to the user. Consequently, the user may organize documents with SONIA and also browse the documents in any subcollection generated with the system *at the same time*. Hence, the ability to actually browse documents is well integrated with the tools for organization that SONIA provides, helping to enable a more topical level of document browsing. For example, Figure 10.9 shows one such Web page generated by SONIA for the "Moons and Rings" subcollection.

Say that the user now wishes to find more articles related to the "Moons and Rings" of Saturn. By selecting this node in the hierarchy and asking the system to

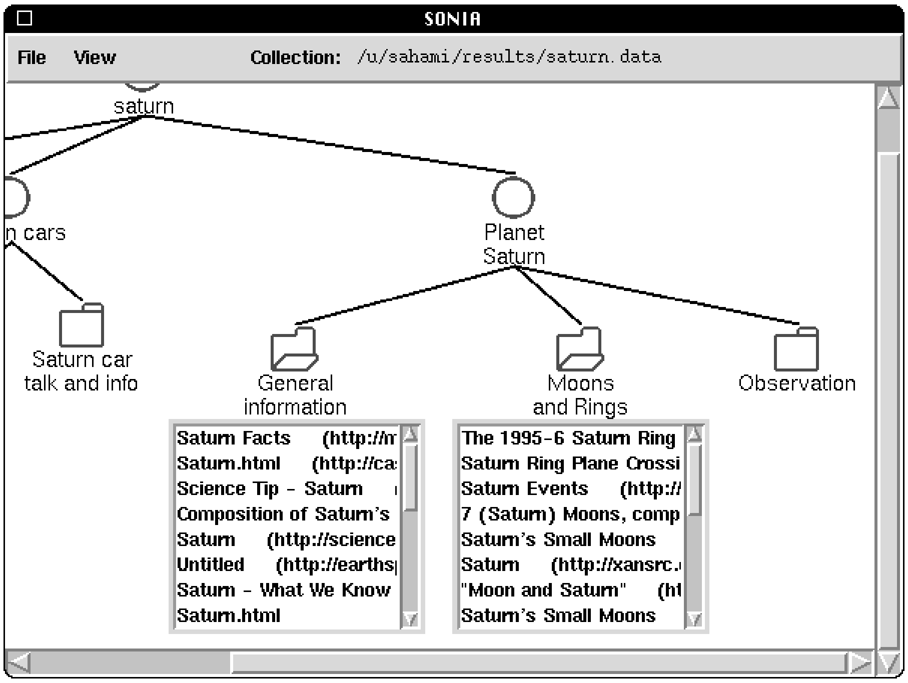

Figure 10.8: SONIA display focusing on the sub-clusters of the "Planet Saturn" node.

saturn	system	solar
saturn's	voyager	tethys
moons	image	cassini
rings	planet	km
ring	pan	atlas
moon	earth	titan
plane	dione	telescope
jupiter	satellites	gif

Table 10.3: Top 24 suggested query terms for "Moons and Rings" subcollection.

suggest query terms, the user invokes the descriptor extraction module of SONIA. This module then displays the top 50 terms related to this collection of documents. We show the top 24 of these terms in Table 10.3. Note that this list of terms includes the specific names of several of Saturn's moons (e.g., Pan, Dione, Tethys, Atlas, and Titan), as well as more general terms for referring to such objects, such as "moons" and "satellites".

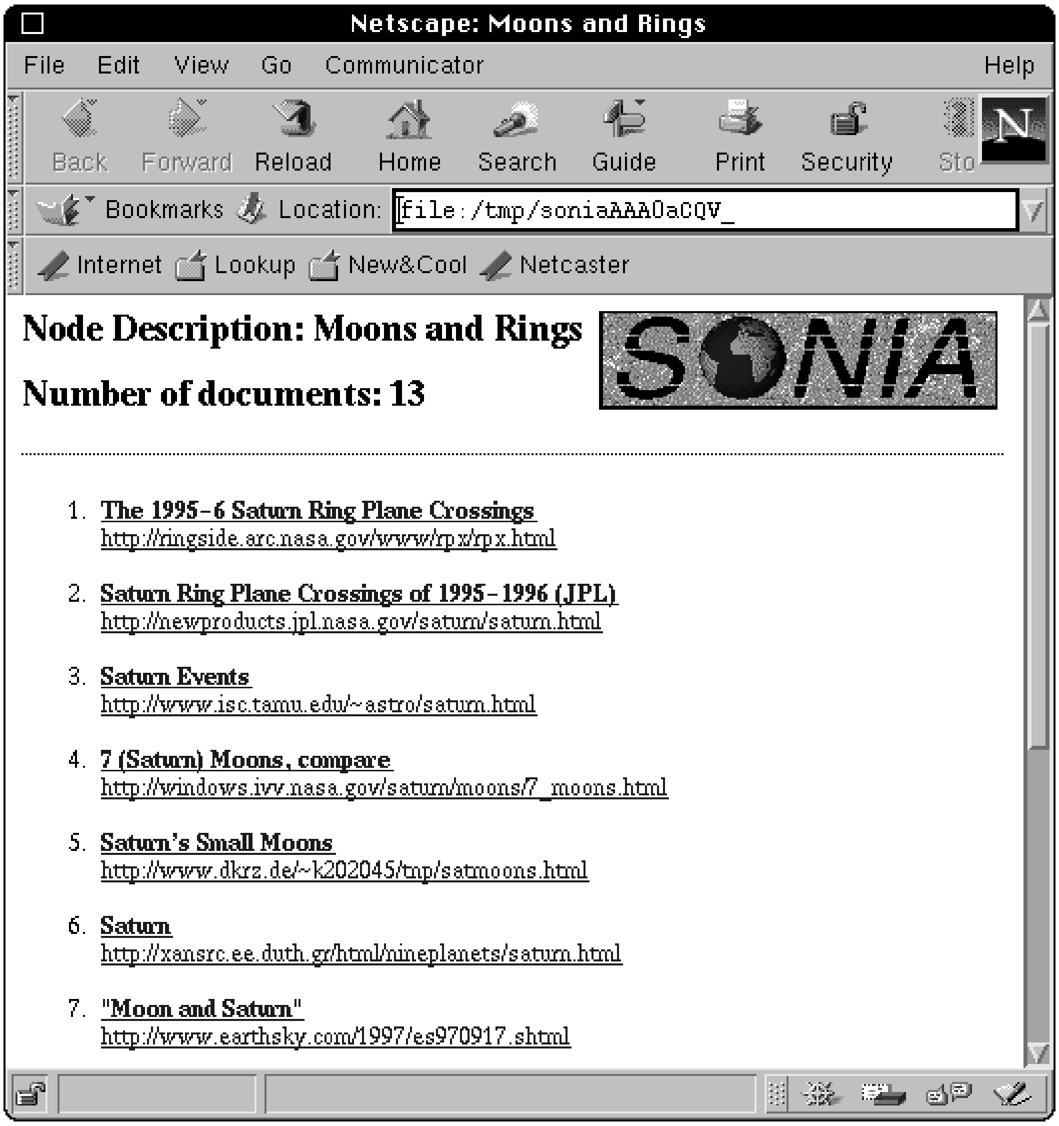

Figure 10.9: A Web page, dynamically generated by SONIA, containing links to documents in the node labeled "Moons and Rings".

With these suggested query terms in hand, the user then issues the follow-up query "saturn satellites and moons" to the system, asking for 50 URLs from the *Excite* search service. As before, this request is serviced using the InfoBus protocol. The crawler in SONIA is capable of retrieving 40 actual web pages from the 50 URLs returned by *Excite*. Rather than organizing these documents from scratch, however, the user makes use of the currently existing topic hierarchy to classify the new incoming documents. These documents are classified into the topic hierarchy by inducing a pachinko machine classifier, as detailed in Section 10.3.5. The process of inducing a hierarchical classification scheme and classifying the new documents takes approximately 20 seconds.

A close inspection of the results of this classification reveals that only one document is not placed somewhere in the "Planet Saturn" subtree. This Web page, which contains a problem set from an astronomy class, would most likely belong in the "Moons and Rings" topic. However, it is mistakenly placed in the "Saturn Car Talk and Info" node, since it contains little actual information about Saturn's moons and rings, and uses very conversational language (akin to many of the "car talk" documents).

In the "Planet Saturn" subtree, three documents are placed in the "General Information" topic, and all these documents do in fact appear to belong in this category, as they contain information predominantly about the planet Saturn itself and only in passing refer to its satellites. The remaining 36 documents retrieved in response to the query are placed in the "Moons and Rings" class. Not surprisingly, 33 of these documents are in fact discussing the moons and rings of Saturn. Two of the documents are about the moons of the planet Jupiter and are arguably classified into the best topic in the hierarchy for them. The last document contains information on the moons of both Jupiter and Saturn, and again seems to be appropriately classified, given the classes that exist in the hierarchy. Thus, it seems that SONIA is quite effective at classifying documents into a hierarchy formed via clustering, making only one error in classifying 40 documents in the example above—a 97.5% accuracy rate. Still, we point out that it would be useful for a system such as SONIA to detect when incoming documents may not fit nicely into any existing topic. We believe that this direction is especially promising for future work.

If more documents were misclassified in the example above (for example, into nodes not in the "Planet Saturn" subtree), it might be argued that the user would not look at these articles if he or she only focused on the results of the "Planet Saturn" subtree. This point becomes less significant, however, when we recognize the vast quantity of relevant documents that a user would never see on a subject because they are not in digital format, have not been indexed, etc. In the context of large information repositories, such as Digital Libraries and the Web, the ability to get query results with high *precision* is generally much more important that being able to *recall* all possibly relevant documents. Thus, for classification, it is arguable that we should generally care more about filtering out non-relevant information than making sure we properly classify all relevant documents. However, we do not pursue this conjecture further here.

10.4.2 Usage Scenario Two

In the second usage scenario, we consider the case where a user makes use of SONIA to help him organize the files in his computer's file system.[2] The user begins with 66 text files that he wishes to organize. Being a graduate student in Computer Science who has been engaged in a recent job search, this user's document set contains both job-related documents (cover letters, various resumes, and letters of recommendation written for former students who are also conducting job searches), as well as some material related to courses (which have recently been taken or taught by the user).

In loading these raw text files into SONIA, the documents are first parsed to produce a vector representation, and then both Zipf's Law-based feature selection and stop word elimination are applied. Since the documents are stored on local disk, there are no network delays in loading this data. Thus, the process of parsing all the documents and applying feature selection, which reduces the feature set from approximately 3500 to just under 900 features, is performed in 20 seconds (again, on

[2]In fact, the particular user in this study is the author of this dissertation, who used SONIA to organize some of the actual documents that had been stored on his personal computer.

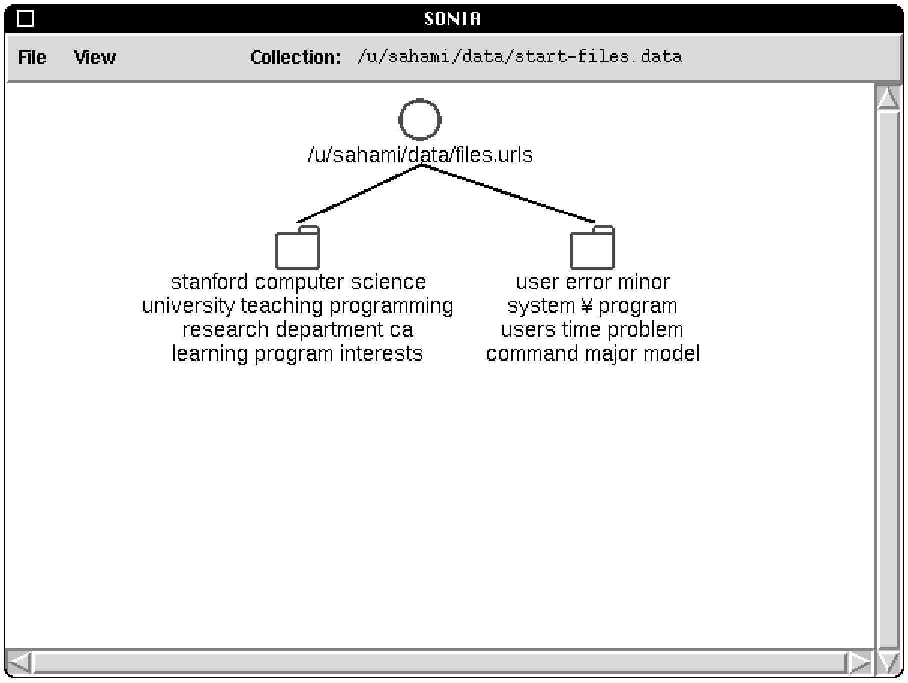

Figure 10.10: SONIA display after initial clustering of the user's file system.

Extracted Descriptors	Sample Document Titles	Plausible Topics	No. Docs
stanford computer science university teaching programming research department ca learning program interests	New_Resume cover-letter-education Andy-Reference	Job related	49
user error minor system • program users time problem command major model	CS147-paper1 psych251-week3 GradingCriteria2	Class related	17

Table 10.4: Results of initially clustering the file system collection.

a heavily loaded Sun SPARC ULTRA 2).

To initially organize these documents, the user decides to cluster the collection into two groups, since there are at least two major themes present. With such a small collection, clustering takes less than 10 seconds to perform. In Table 10.4, we give an overview of these clusters. Also, the SONIA interface showing these two initial clusters is presented in Figure 10.10.

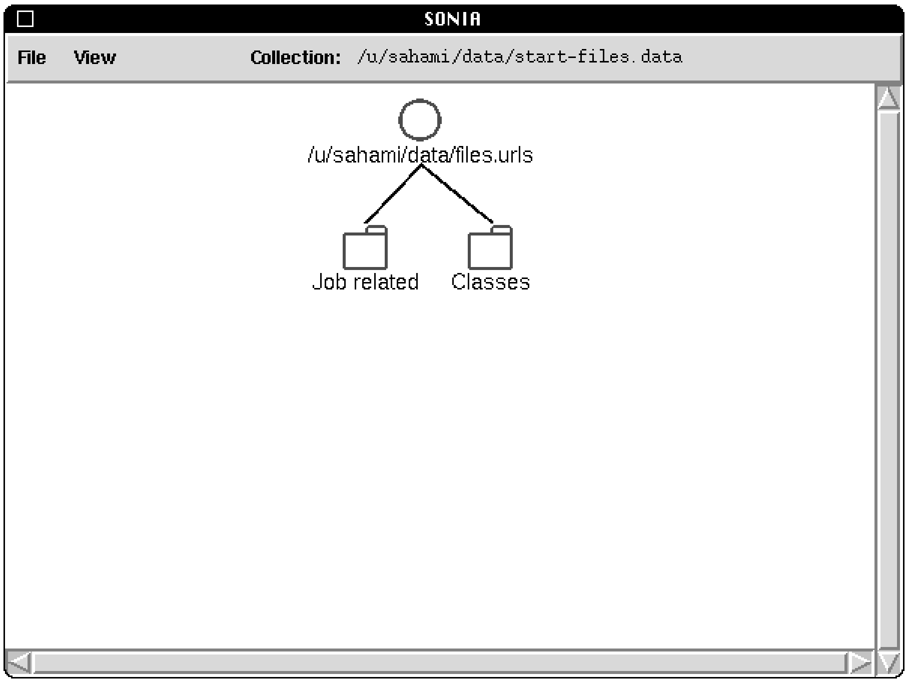

Figure 10.11: SONIA display after naming of the initial file system clusters.

From the results seen in Table 10.4, it is apparent that SONIA can effectively distinguish the job-related document from those pertaining to academic courses. For example, the job-related cluster is characterized by descriptive words such as "stanford", "computer", "science", "teaching", and "research", which are very indicative of the user's background as a graduate student in Computer Science at Stanford University, and his desire to obtain a job involving teaching and research. Also, descriptive terms from the class-related cluster, such as "user" and "model" are reflective of the fact that the user's classes, CS 147 and Psychology 251, are about human-computer interaction and computational models of human learning, respectively. Terms such as "program", "major", "minor", and "error" are also seen in the class-related cluster, reflecting the fact that the grading criteria for the classes the user is teaching measure students' performance by the number of major or minor errors that are made on their programming assignments.

A close examination of the results reveals that the only categorization errors made by the clustering algorithm are two letters of reference (which are clearly job-related)

Extracted Descriptors	Sample Document Titles	Plausible Topics	No. Docs
computer stanford science programming university software ca teaching learning resident responsibilities dormitory	NewResume Resume-newest CurriculumVitae	Resumes	10
stanford computer research university science interests department information teaching consideration learning machine	letter-MIT coverletter-Brown cover-_U_Michigan	Cover Letters	21
stanford computer program science class programming university teaching student section students eric	Ref-Kleper Phil.rec referenceJen	References for others	20

Table 10.5: Results of clustering the "Job related" subcollection.

Extracted Descriptors	Sample Document Titles	Plausible Topics	No. Docs
user system • users command problem model information printer task provided knowledge	CS147-paper1 psych251-week3 GroupPaper	My Courses	9
error minor major errors properly program time array grading face smiley bar	GradingCriteria2 GC1 error-criteria	Grading Criteria	6

Table 10.6: Results of clustering the "Classes" subcollection.

that are incorrectly grouped with the class-related materials. Still, even with these errors, SONIA achieves quite an acceptable level of performance—97.0% accuracy. This result is also notable since the document collection is very skewed toward job-related information, containing 51 such documents, whereas there are only 15 class-related ones. Still, SONIA finds these two major themes, rather than simply trying to produce two convoluted clusters that are of more equal size (as some clustering algorithms using mixture modeling are prone to do [87]). Furthermore, the interactive nature of SONIA's interface makes it quite easy for the user to simply move the two errant documents into the proper folder. The user can then name these two subgroups of documents "Job related" and "Classes", respectively, for easier identifiability. This naming is reflected in the interface shown in Figure 10.11.

Now, say the user wishes to further organize the folder of "Job related" documents, so he clusters this subcollection into three groups. An brief overview of the results of this clustering are given in Table 10.5, where it appears that SONIA has once again successfully identified substructure in this document collection. Here, we find that the three sub-themes of the job-related documents become clear: resumes, cover letters, and job references written for former students. As in the first usage scenario, however, we see that the descriptors automatically extracted for these subgroups contain several overlapping terms, and it is only by viewing the document titles that we can distinguish the major themes of each subgroup. Thus, it seems that our descriptor extraction scheme, while working well at the high levels of the hierarchy, is of more limited utility at finer-grained subtopics lower in the hierarchy.

Performing our usual follow-up examination of the document clustering reveals that only a single reference letter is erroneously included in the same cluster as the cover letters. This yields an accuracy of over 98.0%. Again, the user can use SONIA to simply move this single misclassified document into the correct grouping with minimal effort.

Similarly, the user also decides to cluster the "Classes" subcollection into two groups, trying to separate the documents pertaining to the classes in which he enrolled from those that he taught. The results of this clustering are presented in Table 10.6. Here, the grouping is completely accurate at separating those documents written by the user for classes in which he was enrolled (CS 147 and Psychology 251) from the grading criteria for assignments in the programming classes he taught.

After giving more easily identifiable names to the new document groups formed via clustering, the user produces the hierarchy depicted in Figure 10.12. Note that this hierarchy is essentially a directory structure for helping the user manage this portion of his file system. Indeed, after some time, the user writes several more related documents and it would be natural to use SONIA to classify these newly written documents into the correct place into the file system hierarchy. In accordance with this idea, the user classifies a set of nine new documents (including two cover letters, three new papers for courses he is in, two reference letters for former students of his, an updated resume, and the last grading criteria document for the class he is teaching this quarter) into the current hierarchy. These nine documents were written after the

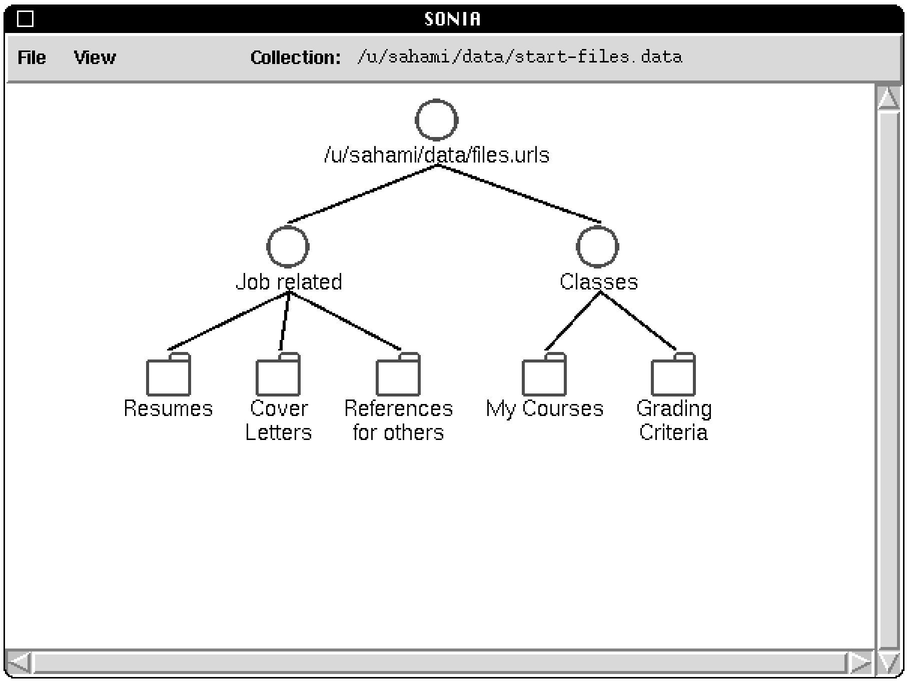

Figure 10.12: SONIA display showing the hierarchy initially constructed by the user.

initial set of documents that were used to generate the hierarchy. SONIA is successful at correctly classifying 100% of these documents into the appropriate classes in the hierarchy using the pachinko machine classification scheme. This shows that SONIA is also effective at helping a user maintain his hierarchical file organization as new documents are being introduced into the system.

After classifying these new documents, the user finally wishes to refine the hierarchy further by differentiating between the different classes he is taking. Thus, he chooses to cluster the "My Courses" folder into two groups (reflecting the number of courses he's been working on most recently). In this case, SONIA distinguishes the two classes with complete accuracy, correctly grouping together all CS 147 papers in one folder and all Psychology 251 papers in another. This final hierarchy (after the "CS147" and "Psych251" folders are named by the user) is shown in Figure 10.13. Note that this figure also shows some of the document titles contained in the "Cover Letters" folder. Here, SONIA marks the newly classified documents with an asterisk (*) at the beginning of their titles to differentiate them from the documents which

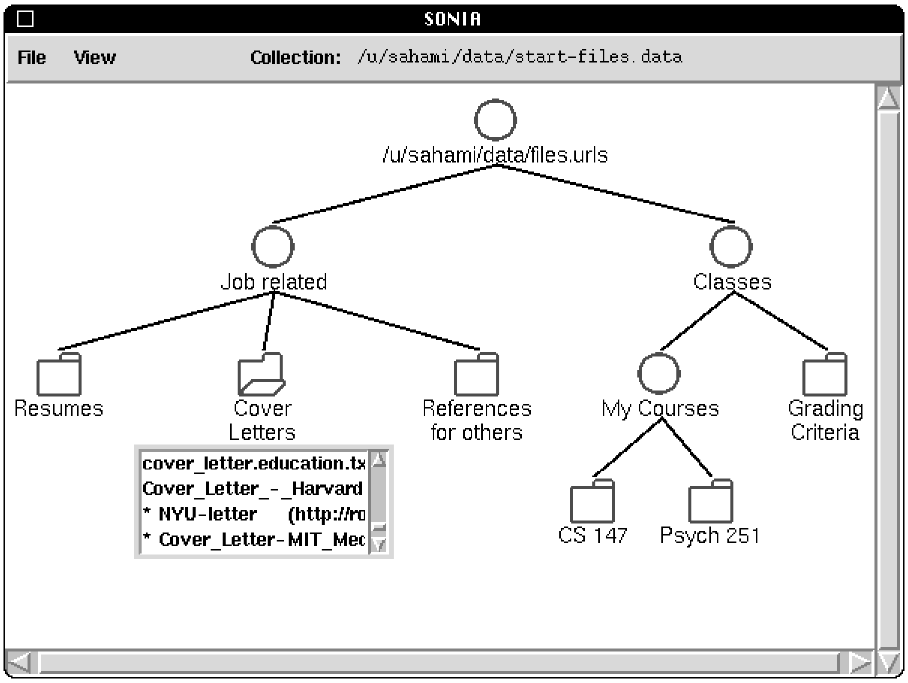

Figure 10.13: SONIA display highlighting the newly classified documents (marked with asterisks) in the "Cover Letters" folder.

previously populated the hierarchy. In this way, the user can quickly visually distinguish the new documents which have been automatically classified into a hierarchy from those that were there previously.

This usage scenario shows that SONIA is effective not only for helping users search for and manage information on the World Wide Web, but also for helping users organize information in much more personal environments, such as the documents on their desktop. Thus, we believe that a system such as SONIA has the potential to be an important component in an intelligent operating system that provides methods for helping users automate their file system management.

10.5 Conclusions

We have presented SONIA, a system that provides the ability to organize document collections into hierarchical categorization schemes, using a variety of machine learning techniques. SONIA is an operational system currently integrated into the Stanford Digital Libraries Project testbed via the InfoBus communications protocol. We have shown that SONIA can effectively help users find and keep track of relevant information in large information spaces by utilizing its automated organizational capabilities. Moreover, by emphasizing the use of hierarchies in document organization, we can leverage users' familiarity with existing hierarchical topical organization schemes used on the World Wide Web (e.g., *Yahoo!*) and personal computer file systems. In this way, we hope to allow users to quickly construct their own personalized and extensible hierarchies of categories, as well as apply SONIA in a variety of other related applications, such as organizing Web page bookmarks.

In future work, we hope to further develop several of the technical modules in SONIA. Most notably, we seek to find better means for descriptor extraction which are robust across a variety of granularity levels in the document hierarchy. Also, we would like to address the issue of clustering or classifying documents into multiple topics in the hierarchy, when appropriate. We believe that basing our work on probability theory can help take a first step in this direction by allowing documents to be included in multiple topics whose probability is close to the maximally probably category. Still, much work is needed to further formalize and implement this idea.

Chapter 11

Conclusions and Future Work

To recap, we briefly summarize some of the results presented in this dissertation in Section 11.1. We also highlight the major themes which have arisen in the course of this work. Then, in Section 11.2 we consider avenues for further research into the use of machine learning in information access. Specifically, we point our several particular issues that have come up in the course of our work that still need to be addressed. Finally, we consider broader research questions aimed at making use of the rich non-textual information which is becoming a more integral part of on-line documents.

11.1 Where Have We Been?

Beginning with the topic of document clustering in Chapters 5 and 6, we first showed that standard methods for clustering via mixture modeling are prone to problems of converging to local maxima in the large parameter spaces of text domains. Furthermore, since it appears that there are many local maxima in these spaces, the clustering results obtained by mixture modeling (even though they were substantially improved through the use of feature selection) still did not appear adequate for use in an end-user system such as SONIA.

Instead, in Chapter 6, we developed a more successful method for clustering which relies on a novel form of probabilistic smoothing based on geometric, rather than arithmetic, means. We also showed that this similarity measure outperforms many of the traditional document similarity measures used in IR for the clustering task. Moreover, in the formulation of our new similarity score, we were able to provide a new interpretation of the commonly used cosine coefficient, thereby helping to shed some new light on an old method. In this way, we hope to have taken some initial steps toward a better understanding of document clustering criteria, and helped point out new directions for further improvements.

Turning to the topic of document classification, we began by developing a new method for feature selection in Chapter 7. In this work, we showed that, for text domains, we could successfully reduce the number of features used in classifier induction to as little as 5% of its original size, and often produce more accurate classification models as a result. Through such feature selection, we reduced the number of parameters that needed to be estimated during the classifier induction process, and thus introduced a bias (i.e., selecting a subset of features) that helped reduce the overall variance associated with the model parameters. Thus, it appears that this way of addressing the bias/variance trade-off can be quite successful in text domains. Moreover, we pointed out that our feature selection method is entirely general, by giving examples of its efficacy in non-text domains as well.

In Chapter 8, we then turned directly to the question of Bayesian classifier induction. Beginning with the working assumption that many feature dependencies exist in text domains, we first characterized the spectrum of feature dependence by defining k-dependence Bayesian classifiers. We then presented an efficient algorithm for the induction of such classifiers and showed that they did indeed provide more accurate classification models for text than models that did not account for feature dependencies. As in the case of feature selection, we also showed the generality of our KDB algorithm on non-textual domains. The bias/variance trade-off once again was an important theme of this work, as we saw that allowing for too great a degree of dependence (usually, $k > 2$), led to a degradation in classifier performance. This seems to show that while the expressivity of the classifier is increased (i.e., less biased toward simple models), any benefits from building a more complex model of the domain are quickly swamped by the exponential increase in the number of model parameters which must be estimated from limited data (and, hence, have high variance).

More generally, we point out the importance of variance control in parameter estimation for modeling text. Consider the fact that for text domains, the ratio of model parameters to the amount of training data available is much higher than in many previously addressed machine learning domains (e.g., the UC Irvine data sets). As a result, it is important to consider what biases may be effective in helping control the variance associated with estimating so many model parameters. For example, the success of the Naive Bayesian classifier can probably be explained more due to its restrictive bias that provides an implicit variance control than for its real efficacy in modeling text. As further evidence of this, we note that when we attempted to use limited dependence Bayesian classifiers to model text domains *without* first performing feature selection we often saw no classification accuracy improvements. This would seem to indicate that any added expressivity from using a richer model structure (which we intuitively believe is more realistic for text) was simply outweighed by the increase in parameter variance.

To further corroborate the need for variance control, we point out that Support Vector Machines (which provide an explicit variance control mechanism) have very recently shown excellent results in text domains [82, 48]. Furthermore, even our clustering results show that, by making use of strong parameter smoothing techniques (another variance control method), we can often obtain superior results. Indeed, as the data mining community looks at ways of constructing algorithms that can scale to huge datasets, those working in text domains must simultaneously think of methods that can scale to enormous parameter spaces and the resulting variance issues associated with them.

Finally, in Chapter 9, we addressed the issue of classifying documents into a hierarchy of topics. We showed that by developing a classification scheme that directly makes use of the hierarchical topic structure, both the features that are examined and the feature dependencies that are modeled can be focused on the most salient aspects of the data in different parts of the document space. In this way, we can better capture contextual information between topics in our hierarchical classifier, and thus build a series of more localized (and, consequently, more accurate) classification models for documents.

Moreover, by developing our classification methods in the same probabilistic framework as our feature selection algorithm, we gained a better understanding of how these different components interact when they are used in combination. For example, we saw that by not making use of probabilistic conditioning information during the feature selection process, we are less likely to eliminate redundant features. Nevertheless, when we subsequently employ limited dependence Bayesian classifiers, which can account for the existing feature dependencies, the problem of redundant features can be dealt with effectively during induction. Thus, combining these methods creates a beneficial synergy that can easily be understood in the probabilistic framework.

Also, in our hierarchical classification scheme, the use of feature selection and limited dependence Bayesian classifiers can be interpreted as building a probabilistic model which has localized feature interaction regions. These regions are defined by the classes in the topic hierarchy. The feature selection algorithm simply selects the most probabilistically relevant features in each region. The limited dependence Bayesian classifier then separately models the dependencies that exist among each subgroup of features.

In conclusion, we point out that while many of the machine learning tools presented in this dissertation have been successfully incorporated into SONIA (as seen in Chapter 10), this system is not the end goal. Rather, it is merely one incarnation of an intelligent information access system using the machine learning tools we have developed here. We hope that there will be others to follow.

11.2 Where Are We Going?

While we hope to have contributed to the improvement of information access systems through the development of machine learning tools geared toward helping users better find and organize on-line information, there is a great deal more work along these lines that needs to be done.

Beginning with our clustering work, we would like to gain a better understanding of the similarity score developed in Chapter 6 from a Bayesian perspective. Moreover, we have only recently uncovered ties between our measure and related work using geometric means of probabilities in the pattern recognition community. We believe that by building stronger ties between our work and these other measures, such as the Bhattacharyya distance, we can not only gain more insight into our particular similarity measure, but hopefully also reveal some more of the important issues pertaining to parameter estimation in high-dimensional domains such as text.

Also along these lines, we would like to explore the use of our similarity measure developed for clustering in supervised classification problems. We believe that our strong initial results in clustering give good evidence that this similarity measure may also be quite effective for classification tasks.

There are also several issues lying at the boundary of clustering and classification which deserve further exploration. Foremost among these is the detection of new topics that do not neatly fit into an existing classification scheme. This area has the potential to benefit from combining previous work in topic detection [1, 176], with new ideas in clustering. Indeed, this issue will become more important as the domain of digital libraries moves from the classic topic structure of existing libraries and into the realm of organizing the quickly evolving body of information on the World Wide Web.

In a similar vein, the notion of automatically merging class hierarchies is another potentially fruitful research area. In this problem, we consider how two separately built hierarchies, which may contain some overlapping topics and documents, may be successfully merged into a single coherent topic hierarchy. If we could adequately address this problem, we could help users elevate their information sharing beyond individual documents (as is commonly done currently), and move into the realm of sharing their large directories of information. We believe that solving this problem will also be a critical issue in the building of large digital libraries through federating existing information repositories.

Yet another area which we believe has great potential is the incorporation of non-textual features, such as document metadata (e.g., author, publisher, etc.), into document clustering and classification tasks. Indeed, initial work along these lines in the context of e-mail classification has already shown that the use of such non-textual features can have a significant impact on classification accuracy [140]. As the current push in the Digital Libraries community toward having more document metadata succeeds in making such non-textual features available for more on-line documents in the future, we believe that methods that can effectively harness this information will have significant advantages over methods that cannot.

While non-textual metadata is one rich source of additional information, methods for incorporating information from other media (e.g., images) into the clustering and classification process also need to be developed. This is especially true given the growing amount of multi-media information being used on Web pages. We hope that by tackling this problem, we can make systems such as SONIA much more applicable to multi-media information repositories.

In looking at the broader scope of a user's need to find and organize relevant information, we would like to consider the development of autonomous software agents which can be directed to scour the Web to find documents related to particular topics in a user's topic hierarchy. We believe that it is possible to make use of the machine learning technologies developed here to help determine the degree of relevance various documents on the Web may have to users. Moreover, to deal with proprietary information, it would be beneficial to incorporate utility models into our work so as to enable such agents to make decision theoretically justified choices regarding, for example, when to pay for proprietary, but potentially highly relevant, information on behalf of a user, or when the agent must consider other cost-sensitive factors when making retrieval decisions.

Finally, we would like to stress the full generality of the machine learning components developed in SONIA, by using the system for exploratory data analysis and data mining in non-textual domains. As our results from Chapters 7 and 8 point out, we believe that the technology incorporated into SONIA has much wider potential than just the organization of textual information. Still, this is an area that needs much work, especially in the identification of domains where such interactive tools could have a large impact.

Indeed, the scope of future work stemming from such a rich area as the use of machine learning in information access is potentially limitless. If we consider research as the on-going search for knowledge and understanding, then perhaps we could best conclude by simply observing that "the point of the journey is not to arrive" [126].

Bibliography

[1] Allan, J., Carbonell, J. G., Doddington, G., Yamron, J., and Yang, Y. Topic detection and tracking pilot study final report. In *Proceedings of the Broadcast News Transcription and Understanding Workshop* (1998).

[2] Allan, J., Leouski, A. V., and Swan, R. C. Interactive cluster visualization for information retrieval. Tech. Rep. IR-116, Uni. of Mass., Amherst, Center for Intelligent Information Retrieval, 1997.

[3] Allen, R. B., Obry, P., and Littman, M. An interface for navigating clustered document sets returned by queries. In *Proceedings of ACM SIGOIS* (1993), pp. 166–171.

[4] Almuallim, H., Akiba, Y., and Kaneda, S. An efficient algorithm for finding optimal gain-ratio multiple-split tests on hierarchical attibutes in decision tree learning. In *Thirteenth National Conference on Artificial Intelligence* (1996), pp. 703–708.

[5] Almuallim, H., and Dietterich, T. G. Learning with many irrelevant features. In *Ninth National Conference on Artificial Intelligence* (1991), MIT Press, pp. 547–552.

[6] AltaVista. Digital Equipment Corporation internet search service. http://www.altavista.digital.com/, 1995.

[7] Apte, C., Damerau, F., and Weiss, S. M. Automated learning of decision rules for text categorization. *Transactions of Office Information Systems 12*, 3 (1994). Special Issue on Text Categorization.

[8] Balabanovic, M. An adaptive web page recommendation service. In *Proceedings of the First International Conference on Autonomous Agents* (1997).

[9] Balabanovic, M., and Shoham, Y. Fab: Content-based, collaborative recommendation. *Communications of the ACM 40*, 3 (1997).

[10] Baldonado, M., Chang, C.-C. K., Gravano, L., and Paepcke, A. The Stanford digital library metadata architecture. *International Journal of Digital Libraries 1*, 2 (1997).

[11] Baldonado, M. Q. W., and Winograd, T. SenseMaker: An information-exploration interface supporting the contextual evolution of a user's interests. In *Proceedings of CHI* (1997).

[12] Bhattacharyya, A. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc. 35* (1943), 99–109.

[13] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. Occam's razor. *Information Processing Letters 24* (1987), 377–380.

[14] Boutilier, C., Friedman, N., Goldszmidt, M., and Koller, D. Context-specific independence in bayesian networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence* (1996), pp. 115–123.

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

[16] Buntine, W. Learning classification trees. *Statistics and Computing 2* (June 1992), 63–73.

[17] Buntine, W. Operations for learning with graphical models. *Journal of Artificial Intelligence Research 2* (1994), 159–225.

[18] Callan, J. Characteristics of text. http://hobart.cs.umass.edu/~allan/cs646/char_of_text.html, 1997.

[19] Callan, J., Croft, W. B., and Broglio, J. TREC and TIPSTER experiments with INQUERY. *Information Processing and Management 32* (1995), 327–332.

[20] Caruana, R., and Freitag, D. Greedy attribute selection. In *Machine Learning: Proceedings of the Eleventh International Conference* (1994), Morgan Kaufmann.

[21] Charniak, E. *Statistical Language Learning.* MIT Press, Cambridge, MA, 1993.

[22] Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D. AutoClass: a bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning* (1988), Morgan Kaufmann Publishers, Inc., pp. 54–64. Also appears in *Readings in Machine Learning* edited by J. Shavlik and T. Dietterich.

[23] Cheeseman, P., and Stutz, J. Bayesian classification (AutoClass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. AAAI Press/MIT Press, 1996.

[24] Chen, S. F., and Goodman, J. T. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics* (1996), pp. 310–318.

[25] Chickering, D. M. Learning bayesian networks is NP-complete. In *Lecture Notes in Statistics* (1995).

[26] Chow, C., and Liu, C. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory 114* (1968), 462–467.

[27] Cohen, W. W. Learning rules that classify e-mail. In *Proceedings of the 1996 AAAI Spring Symposium on Machine Learning in Information Access* (1996).

[28] Cohen, W. W. Learning trees and rules with set-valued features. In *AAAI-96: Proceedings of the Thirteenth National Conference on Artificial Intelligence* (1996).

[29] Cohen, W. W., and Singer, Y. Context-sensitive learning methods for text categorization. In *Proceedings of the 19th Annual ACM SIGIR Conference* (1996).

[30] Cooper, G. F. Probabilistic inference using belief networks is NP-Hard. Tech. Rep. KSL-87-27, Stanford Knowledge Systems Laboratory, 1987.

[31] Cooper, G. F., and Herskovits, E. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning 9* (1992), 309–347.

[32] Cooper, W. S. The formalism of probability theory in IR: A foundation for an encumbrance? In *Proceedings of the 17th Annual International ACM/SIGIR Conference* (Dublin, Ireland, 1994), pp. 242–247.

[33] Cooper, W. S., and Maron, M. E. Foundations of probabilistic and utility-theoretic indexing. *Journal of the Association for Computing Machinery 25*, 1 (1978), 67–80.

[34] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. *Introduction to Algorithms.* McGraw-Hill, 1990.

[35] Cover, T. M., and Thomas, J. A. *Elements of Information Theory.* Wiley, 1991.

[36] Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S. Learning to extract symbolic knowledge from the world wide web. In *AAAI-98: Proceedings of the Fifteenth National Conference on Artificial Intelligence* (1998).

[37] Crawford, S. L., Fung, R. M., Appelbaum, L. A., and Tong, R. M. Classification trees for information retrieval. In *Machine Learning: Proceedings of the Eighth International Workshop* (1991), Morgan Kaufmann, pp. 245–249.

[38] Cristianini, N., Shawe-Taylor, J., and Sykacek, P. Bayesian classifiers are large margin hyperplanes in a Hilbert space. In *Machine Learning: Proceedings of the Fifteenth International Conference* (1998).

[39] Cutting, D. R., Karger, D. R., Pederson, J. O., and Tukey, J. W. Scatter/Gather: a cluster-based approach to browsing large document collections. In *Proceedings of ACM/SIGIR* (1992), pp. 318–329.

[40] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, 6 (1990), 391–407.

[41] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B 39* (1977), 1–39.

[42] Dietterich, T. G., and Shavlik, J. W., Eds. *Readings in Machine Learning*. Morgan Kaufmann Publishers, Inc., 1990.

[43] Domingos, P., and Pazzani, M. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning 29* (1997), 103–130.

[44] Dougherty, J., Kohavi, R., and Sahami, M. Supervised and unsupervised discretization of continuous features. In *Machine Learning: Proceedings of the Twelfth International Conference* (July 1995), A. Prieditis and S. Russell, Eds., Morgan Kaufmann Publishers, Inc.

[45] Draper, D., and Hanks, S. Localized partial evaluation of belief networks. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI '94)* (1994), pp. 170–177.

[46] Duda, R., and Hart, P. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[47] Dumais, S. T. Latent semantic indexing (LSI) and TREC-2. In *Proceeding of the Second Text REtrieval Conference (TREC-2)*, D. K. Harman, Ed. National Institute of Standards and Technology, 1993, pp. 105–115.

[48] Dumais, S. T., Platt, J., Heckerman, D., and Sahami, M. Inductive learning algorithms and representations for text categorization. In *CIKM-98: Proceedings of the Seventh International Conference on Information and Knowledge Management* (1998).

[49] Ezawa, K. J., and Schuermann, T. Fraud/uncollectible debt detection using a bayesian network learning system. In *Uncertainty in Artificial Intelligence* (1995), Elsevier Science, pp. 157–166.

[50] Frakes, W. B. Stemming algorithms. In *Information Retrieval: Data Structures and Algorithms*, W. B. Frakes and R. Baeza-Yates, Eds. Prentice Hall, 1992, pp. 131–160.

[51] Frakes, W. B., and Baeza-Yates, R. *Information Retrieval: Data Structures and Algorithms.* Prentice Hall, 1992.

[52] Friedman, N., Geiger, D., and Goldszmidt, M. Bayesian network classifiers. *Machine Learning 29* (1997), 131–163.

[53] Friedman, N., and Goldszmidt, M. Building classifiers using bayesian networks. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (1996).

[54] Friedman, N., and Goldszmidt, M. Learning bayesian networks with local structure. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence* (1996), pp. 252–262.

[55] Fuernkranz, J., Mitchell, T., and Riloff, E. A case study in using linguistic phrases for text categorization on the WWW. In *Learning for Text*

Categorization: Papers from the 1998 AAAI Workshop (Technical Report WS-98-05) (1998), M. Sahami, Ed.

[56] FUHR, N. Models for retrieval with probabilistic indexing. *Information Processing and Management 25*, 1 (1989), 55–72.

[57] FUHR, N. Probabilistic models in information retrieval. *The Computer Journal 35*, 3 (1992), 243–255.

[58] FUKUNAGA, K. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[59] FUNG, R., AND DELFAVERO, B. Applying bayesian networks to information retrieval. *Communications of the ACM 38*, 3 (1995), 42–48.

[60] GEIGER, D. An entropy-based learning algorithm of bayesian conditional trees. In *Uncertainty in Artificial Intelligence* (1992), Elsevier Science, pp. 92–97.

[61] GEMAN, S., BIENENSTOCK, E., AND DOURSAT, R. Neural networks and the bias/variance dilemma. *Neural Computation 4* (1992), 1–48.

[62] GIROSI, F. An equivalence between sparse approximation and support vector machines. Tech. Rep. AI Memo 1606, Massachusetts Institute of Technology, 1997. To appear in *Neural Computation.*

[63] GOLDSZMIDT, M., AND SAHAMI, M. A probabilistic approach to full-text document clustering. Tech. Rep. ITAD-433-MS-98-044, SRI International, 1998.

[64] GOOD, I. J. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods.* M.I.T. Press, 1965.

[65] HARMAN, D. K., AND VOORHEES, E. M., Eds. *Proceedings of the Sixth Text REtrieval Conference (TREC-6).* National Institute of Standards and Technology, 1997.

[66] Harmon, D. Ranking algorithms. In *Information Retrieval: Data Structures and Algorithms*, W. B. Frakes and R. Baeza-Yates, Eds. Prentice Hall, 1992, pp. 363–392.

[67] Hearst, M. Interfaces for searching the web. *Scientific American* (March 1997).

[68] Hearst, M., and Pedersen, J. Revealing collection structure through information access interfaces. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (1995), pp. 2047–2048. Video tape.

[69] Hearst, M. A. The use of categories and clusters in information access interfaces. In *Natural Language Information Retrieval*, T. Strzalkowski, Ed. Kluwer, 1999. In press.

[70] Hearst, M. A., Karger, D. R., and Pederson, J. O. Scatter/Gather as a tool for the navigation of retrieval results. In *Proceedings of AAAI Fall Symposium on Knowledge Navigation* (1995).

[71] Hearst, M. A., and Pederson, J. O. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *Proceedings of ACM/SIGIR* (1996).

[72] Heckerman, D., Geiger, D., and Chickering, D. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning 20* (1995), 197–243.

[73] Heckerman, D., and Platt, J. Personal communication, 1998.

[74] Heckerman, D. E., Horvitz, E. J., and Nathwani, B. N. Toward normative expert systems: Part i. The Pathfinder project. *Methods of Information in Medicine 31* (1992), 90–105.

[75] Heckerman, D. E., and Nathwani, B. N. Toward normative expert systems: Part ii. Probability-based representations for efficient knowledge acquisition and inference. *Methods of Information in Medicine 31* (1992), 106–116.

[76] HERSH, W. R., BUCKLEY, C., LEONE, T. J., AND HICKAM, D. H. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual ACM SIGIR Conference* (1994), pp. 192–201.

[77] HULL, D. A. The TREC-6 filtering track: Description and analysis. In *Proceeding of the Sixth Text REtrieval Conference (TREC-6)*, D. K. Harman and E. M. Voorhees, Eds. National Institute of Standards and Technology, 1997.

[78] INFOSEEK. Internet directory and query service. http://www.infoseek.com/, 1995.

[79] INFOSEEK. Aptex categorizes more than 700,000 web sites for infoseek. http://info.infoseek.com/doc/PressReleases/hnc.html, 1996.

[80] JAAKKOLA, T. S., AND HAUSSLER, D. Exploiting generative models in discriminative classifiers. http://www.cse.ucsc.edu/research/ml/papers/Jaakola.ps, 1998.

[81] JOACHIMS, T. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In *Machine Learning: Proceedings of the Fourteenth International Conference* (1997), Morgan Kaufmann, pp. 143–151.

[82] JOACHIMS, T. Text categorization with support vector machines: Learning with many relevant features. Tech. Rep. LS8-Report, Universitaet Dortmund, 1997.

[83] JOACHIMS, T., FREITAG, D., AND MITCHELL, T. WebWatcher: A tour guide for the world wide web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (1997).

[84] JOHN, G., KOHAVI, R., AND PFLEGER, K. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference* (1994), Morgan Kaufmann, pp. 121–129.

[85] John, G. H., and Langley, P. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI '95)* (1995), pp. 338–345.

[86] Kailath, T. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology COM-15*, 1 (February 1967), 52–60.

[87] Kearns, M., Mansour, Y., and Ng, A. Y. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence* (1997), pp. 282–293.

[88] Kira, K., and Rendell, L. A. The feature selection problem: Traditional methods and a new algorithm. In *AAAI-92: Proceedings of the Tenth National Conference on Artificial Intelligence* (1992), MIT Press, pp. 129–134.

[89] Kivinen, J., and Warmuth, M. Exponentiated gradient versus gradient descent for linear predictors. Tech. Rep. UCSC-CRL-94-16, Basking Center for Computer Engineering and Information Sciences; University of California, Santa Cruz, 1994.

[90] Kohavi, R. *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. PhD thesis, Stanford University, Computer Science department, 1995.

[91] Kohavi, R., Becker, B., and Sommerfield, D. Improving simple bayes. In *ECML-97: Proceedings of the Ninth European Conference on Machine Learning* (1997).

[92] Kohavi, R., and John, G. Automatic parameter selection by minimizing estimated error. In *Machine Learning: Proceedings of the Twelfth International Conference* (1995), A. Prieditis and S. Russell, Eds., Morgan Kaufmann Publishers, Inc.

[93] Kohonen, T. *Self-Organizing Maps*. Springer, 1995.

[94] KOLLER, D., AND SAHAMI, M. Toward optimal feature selection. In *Machine Learning: Proceedings of the Thirteenth International Conference* (1996), Morgan Kaufmann, pp. 284–292.

[95] KOLLER, D., AND SAHAMI, M. Hierarchically classifying documents using very few words. In *Machine Learning: Proceedings of the Fourteenth International Conference* (1997), Morgan Kaufmann, pp. 170–178.

[96] KONONENKO, I. Semi-naive bayesian classifier. In *Proceedings of the Sixth European Working Session on Learning* (1991), Pitman, pp. 206–219.

[97] KOZLOV, A. V., AND SINGH, J. P. Sensitivities: An alternative to conditional probabilities for Bayesian belief networks. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI '95)* (1995), pp. 376–385.

[98] KRISHNAIAH, P. R., AND KANAL, L. N. *Classification, Pattern Recognition, and Reduction in Dimensionality.* Amsterdam: North Holland, 1982.

[99] KULLBACK, S. *Information Theory and Statistics.* Wiley, 1959.

[100] KULLBACK, S., AND LEIBLER, R. A. On information and sufficiency. *Annals of Mathematical Statistics 22* (1951), 76–86.

[101] LANG, K. NewsWeeder: Learning to filter news. In *Machine Learning: Proceedings of the Twelfth International Conference* (1995), Morgan Kaufmann.

[102] LANGLEY, P. *Elements of Machine Learning.* Morgan Kaufmann Publishers, Inc., San Francisco, 1995.

[103] LANGLEY, P., IBA, W., AND THOMPSON, K. An analysis of bayesian classifiers. In *Proceedings of the tenth national conference on artificial intelligence* (1992), AAAI Press and MIT Press, pp. 223–228.

[104] LANGLEY, P., AND SAGE, S. Induction of selective bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence* (1994), Morgan Kaufmann, pp. 399–406.

[105] LAWRENCE, S., AND GILES, C. L. Searching the world wide web. *Science 280*, 5360 (1998), 98–100.

[106] LEWIS, D. D. *Representation and Learning in Information Retrieval.* PhD thesis, Univ. of Massachusetts, Amherst, Department of Computer Science, 1992.

[107] LEWIS, D. D. The TREC-5 filtering track. In *Proceeding of the Fifth Text REtrieval Conference (TREC-5)*, D. K. Harman and E. M. Voorhees, Eds. National Institute of Standards and Technology, 1996, pp. 75–96.

[108] LEWIS, D. D. Naive (Bayes) at forty: The independence assumption in information retrieval. In *ECML-98: Proceedings of the Tenth European Conference on Machine Learning* (1998).

[109] LEWIS, D. D., AND GALE, W. A. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM/SIGIR Conference* (Dublin, Ireland, 1994), pp. 3–11.

[110] LEWIS, D. D., AND RINGUETTE, M. A comparison of two learning algorithms for text categorization. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval* (1994), pp. 81–93.

[111] LEWIS, D. D., SCHAPIRE, R. E., CALLAN, J. P., AND PAPKA, R. Training algorithms for linear text classifiers. In *Proceedings of ACM/SIGIR* (1996).

[112] LIN, X., SOERGEL, D., AND MARCHIONINI, G. A self-organizing semantic map for information retrieval. In *Proceedings of ACM/SIGIR* (1991), pp. 262–269.

[113] MARON, M. E. Mechanized documentation: The logic behind a probabilistic interpretation. *Statistical Association Methods for Mechanized Documentation* (1965), 9–13.

[114] MASAND, B., LINOFF, G., AND WALTZ, D. Classifying news stories using memory based reasoning. In *Proceedings of ACM/SIGIR* (1992), pp. 59–65.

[115] McCallum, A., and Nigam, K. A comparison of event models for naive bayes text classification. In *Learning for Text Categorization: Papers from the 1998 AAAI Workshop (Technical Report WS-98-05)* (1998), M. Sahami, Ed.

[116] McCallum, A., Rosenfeld, R., Mitchell, T., and Ng, A. Improving text classification by shrinkage in a hierarchy of classes. In *Machine Learning: Proceedings of the Fifteenth International Conference* (1998).

[117] Mitchell, T. M. *Machine Learning.* McGraw-Hill, 1997.

[118] Mladenic, D. Turning yahoo into an automatic web-page classifier. In *ECAI-98: Proceedings of the Thirteenth European Conference on Aritficial Intelligence* (1998), pp. 473–474.

[119] Murphy, P. M., and Aha, D. W. UCI repository of machine learning databases. http://www.ics.uci.edu/˜mlearn/MLRepository.html, 1995.

[120] Negroponte, N. *Being Digital.* Vintage Books, 1995.

[121] Nilsson, N. J. *The Mathematical Foundations of Learning Machines.* Morgan Kaufmann Publishers, Inc., 1990. Previously published as: Learning Machines, 1965.

[122] Pazzani, M., and Billsus, D. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning 27* (1997), 313–331.

[123] Pazzani, M., Muramatsu, J., and Billsus, D. Syskill & Webert: Identifying interesting web sites. In *AAAI-96: Proceedings of the Thirteenth National Conference on Artificial Intelligence* (1996).

[124] Pazzani, M. J. Searching for dependencies in bayesian classifiers. In *Proceedings of the fifth International Workshop on Artificial Intelligence and Statistics* (Ft. Lauderdale, FL, January 1995), D. Fisher and H. Lenz, Eds.

[125] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan-Kaufmann, 1988.

[126] Peart, N. Prime mover. Lyrics from the album *Hold Your Fire*, 1987.

[127] Pirolli, P., Schank, P., Hearst, M., and Diehl, C. Scatter/gather browsing communicates the topic structure of a very large text collection. In *Proceedings of CHI* (1996).

[128] Porter, M. F. An algorithm for suffix stripping. *Program 14*, 3 (1980), 130–137.

[129] Quinlan, J. R. Induction of decision trees. *Machine Learning 1* (1986), 81–106. Reprinted in Shavlik and Dietterich (eds.) *Readings in Machine Learning*.

[130] Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., Los Altos, California, 1993.

[131] Rasmussen, E. Clustering algorithms. In *Information Retrieval: Data Structures and Algorithms*, W. B. Frakes and R. Baeza-Yates, Eds. Prentice Hall, 1992, pp. 419–442.

[132] Reuters. Reuters-22173 document collection. Distribution for research purposes has been granted by Reuters and Carnegie Group. Arrangements for access were made by David Lewis. An updated version of this data set, Reuters-21578, is now publically available from David Lewis at `http://www.research.att.com/~lewis.`, 1995.

[133] Riloff, E., and Lehnart, W. Information extraction as a basis for high-precision text classification. *Transactions of Office Information Systems 12*, 3 (1994). Special Issue on Text Categorization.

[134] Robertson, S. E. The probability ranking principle in IR. *Journal of Documentation 33*, 4 (1977), 292–304.

[135] Robertson, S. E., and Sparck-Jones, K. Relevance weighting of search terms. *Journal of the American Society of Information Science 27* (1976), 129–146.

[136] Robertson, S. E., and Walker, S. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the Seventeenth International Conference on Research and Development in Information Retrieval* (1994), pp. 232–241.

[137] Rocchio, J. J. Relevance feedback in information retrieval. In *The SMART Information Retrieval System*, G. Salton, Ed. Prentice Hall, Englewood Cliffs, NJ, 1971, pp. 313–323.

[138] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. In *Parallel Distributed Processing*, J. McClelland and D. Rumelhart, Eds. MIT Press, 1986. Chapter 8.

[139] Sahami, M. Learning limited dependence Bayesian classifiers. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (1996), pp. 335–338.

[140] Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 AAAI Workshop (Technical Report WS-98-05)* (1998), M. Sahami, Ed.

[141] Sahami, M., Hearst, M., and Saund, E. Applying the multiple cause mixture model to text categorization. In *Machine Learning: Proceedings of the Thirteenth International Conference* (1996), Morgan Kaufmann, pp. 435–443.

[142] Sahami, M., Yusufali, S., and Baldonado, M. Q. W. Real-time full-text clustering of networked documents. In *AAAI-97: Proceedings of the Fourteenth National Conference on Artificial Intelligence* (1997), p. 845.

[143] Sahami, M., Yusufali, S., and Baldonado, M. Q. W. SONIA: A service for organizing networked information autonomously. In *Digital Libraries 98: Proceedings of the Third ACM Conference on Digital Libraries* (1998).

[144] SALTON, G. *The SMART Information Retrieval System.* Prentice Hall, Englewood Cliffs, NJ, 1971.

[145] SALTON, G., AND BUCKLEY, C. Term weighting approaches in automatic text retrieval. *Information Processing and Management 24*, 5 (1988), 513–523.

[146] SALTON, G., AND MCGILL, M. J. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company, 1983.

[147] SALTON, G., WONG, A., AND YANG, C. S. A vector space model for automatic indexing. *Communications of the ACM 18* (1975), 613–620.

[148] SCHUETZE, H., HULL, D., AND PEDERSEN, J. A comparison of document representations and classifiers for the routing problem. In *Proceedings of the 18th Annual ACM SIGIR Conference* (1995), pp. 229–237.

[149] SCHUETZE, H., AND SILVERSTEIN, C. A comparison of projections for efficient document clustering. In *Proceedings of ACM/SIGIR* (1997).

[150] SELF, G. Personal communication, 1996.

[151] SELTZER, R., RAY, D. S., AND RAY, E. J. *The AltaVista Search Revolution.* McGraw-Hill, 1998.

[152] SINGH, M., AND PROVAN, G. M. A comparison of induction algorithms for selective and non-selective bayesian classifiers. In *Machine Learning: Proceedings of the Twelfth International Conference* (July 1995).

[153] SINGH, M., AND PROVAN, G. M. Efficient learning of selective bayesian network classifiers. In *Machine Learning: Proceedings of the Thirteenth International Conference* (1996), Morgan Kaufmann, pp. 453–461.

[154] SINGH, M., AND VALTORTA, M. An algorithm for the construction of bayesian network structures from data. In *Uncertainty in Artificial Intelligence* (1993), Elsevier Science, pp. 259–265.

[155] Sparck-Jones, K., and Willett, P., Eds. *Readings in Information Retrieval.* Morgan Kaufmann, 1997.

[156] Spertus, E. Smokey: Automatic recognition of hostile messages. In *Proceedings of Innovative Applications of Artificial Intelligence (IAAI)* (1997), pp. 1058–1065.

[157] Stanfill, C., and Waltz, D. Toward memory-based reasoning. *Communications of the ACM 29*, 12 (1986), 1213–1228.

[158] Stanford Digital Libraries Group. The Stanford digital libraries project. *Communications of the ACM* (April 1995).

[159] Strang, G. *Linear Algebra and its Applications.* Harcourt Brace Jovanovich, 1988.

[160] Sullivan, D. The major search engines. From the *Search Engine Watch Report* available at http://www.searchenginewatch.com/facts/major.html, 1998.

[161] Taylor, C., Michie, D., and Spiegalhalter, D. *Machine Learning, Neural and Statistical Classification.* Paramount Publishing International, 1994.

[162] Timo, H., Samuel, K., Krista, L., and Teuvo, K. WEBSOM - self-organizing maps of document collections. In *Proceedings of WSOM'97 Workshop on Self-Organizing Maps* (1997), pp. 310–315.

[163] Turtle, H., and Croft, W. B. Inference networks for document retrieval. In *Proceedings of the Thirteenth International Conference on Research and Development in Information Retrieval* (1990), pp. 1–24.

[164] Turtle, H. R. *Inference Networks for Document Retrieval.* PhD thesis, Univ. of Massachusetts, Amherst, Department of Computer Science, 1991.

[165] van Rijsbergen, C. J. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation 33* (1977), 106–119.

[166] van Rijsbergen, C. J. *Information Retrieval.* Butterworths, 1979.

[167] van Rijsbergen, C. J., and Jardine, N. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval 7* (1971), 217–240.

[168] Vapnik, V. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

[169] Voorhees, E. M. *The Effectiveness and Efficiency of Agglomerative Hierarchical Clustering in Document Retrieval*. PhD thesis, Cornell University, 1986.

[170] Widrow, B., and Winter, R. G. Neural nets for adaptive filtering and adaptive pattern recognition. *IEEE Computer* (March 1988), 25–39.

[171] Wiener, E., Pedersen, J. O., and Weigand, A. S. A neural network approach to topic spotting. In *Symposium on Document Analysis and Information Retrieval* (1995), pp. 317–332.

[172] Willett, P. Recent trends in hierarchical document clustering: A critical review. *Information Processing and Management 24*, 5 (1988), 577–597.

[173] Yahoo! On-line guide for the internet. http://www.yahoo.com/, 1995.

[174] Yang, Y., and Chute, C. G. An example-based mapping method for text categorization and retrieval. *Transactions of Office Information Systems 12*, 3 (1994). Special Issue on Text Categorization.

[175] Yang, Y., and Pedersen, J. Feature selection in statistical learning of text categorization. In *Machine Learning: Proceedings of the Fourteenth International Conference* (1997), Morgan Kaufmann, pp. 412–420.

[176] Yang, Y., Pierce, T., and Carbonell, J. G. A study on retrospective and on-line event detection. In *Proceedings of the 21st Annual ACM SIGIR Conference* (1998).

[177] Zipf, G. K. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.