Robust Statistical Techniques for the Categorization of Images Using Associated Text

Carl Sable

Submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2003

ABSTRACT

Robust Statistical Techniques for the Categorization of Images Using Associated Text

Carl Sable

The field of text categorization, which aids applications such as browsing, filtering, and search, has experienced a revival due to the vast amounts of unlabeled data available on line and as part of digital collections. Almost all of the literature in the field, however, deals with the categorization of text-only documents. Many of the same techniques can be applied to text associated with multimedia documents to label the multimedia component. My dissertation provides an in-depth exploration of the automatic categorization of images using associated text. This research takes advantage of a corpus I have created containing news documents with embedded captioned images and multiple sets of categories. It turns out that the text and categories associated with images tend to have different properties than those associated with full-length text documents such as e-mails, articles, and web pages. Also, images provide us with an additional type of information; namely, low-level image features. For these reasons, I have achieved success in several areas of research that have previously been problematic, such as combining systems and using NLP techniques to improve performance. Some benefits of this work are demonstrated as part of Columbia's Newsblaster system, which finds, clusters, categorizes, and summarizes news on the web. Newsblaster has already captured

the attention of the public and press; articles about Newsblaster have appeared in sources including The New York Times, USA Today, and Slashdot, and a recent analysis indicates that Newsblaster receives tens of thousands of hits every day.

The research discussed in this dissertation fits into two general paradigms. One paradigm involves research in machine learning techniques. Within this framework, I have developed two text categorization systems that use novel approaches. One of these systems applies a statistical technique known as density estimation to the output generated by another system. Density estimation provides probabilistic confidence measures for predictions and often improves accuracy. The other system, BINS, uses a binning technique to empirically estimate term weights for groups of words that share statistical features in common (instead of estimating term weights for individual words). This enables the system to compute accurate term weights for words with scarce evidence. Other work that fits into this paradigm involves the combination of a set of rules, each of which is very accurate but rarely applicable, with systems to which we can fall back when the rules do not apply.

The second paradigm of research, less common in the literature, involves novel representation of documents. Almost all modern text categorization systems use bag of words approaches, meaning that documents are represented by vectors of weighted words. Neither syntax nor semantics are considered, and no other information is used. For specific categories applying to images, however, I have found that substantial improvement can be obtained using more advanced NLP techniques. I confirm this hypothesis by presenting evidence from experiments with human subjects who have viewed image captions under varying conditions. I then describe an NLP based system that significantly outperforms all standard systems that have been tested for a specific task. Additional work that fits into this paradigm includes the use of low-level image features (e.g. color), alone or in combination with text, to categorize images.

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

First I'd like to thank the members of my committee. Thanks to Shree Nayar for agreeing to be on the committee, and for accepting a date for the defense that I know is closer to the holidays than he wanted. Thanks to Shih-Fu Chang (and his students Seungyup and Ana) for collaboration with research involving image features. Thanks to Vasileios Hatzivassiloglou for assistance over the years; his input has been essential for much of the research discussed in this thesis. Special thanks to Ken Church, whose advice over the past two and a half years has clearly made me a better researcher. More thanks to Ken for two great summer internships and for many great ideas that have influenced my work in general.

Most of all, I'd like to thank my advisor, Kathy McKeown. It has been a pleasure to be her student; the experience has been greatly rewarding in many respects. Due to her guidance, the last five and a half years have been intellectually stimulating and challenging, but rarely overwhelming. All of the research discussed in this thesis has been influenced by her. Additional thanks to Kathy for the freedom to make my own decisions, to pursue internships or stay at Columbia every summer, and to teach every semester that I desired.

I'd also like to thank all of the members of Columbia's NLP group, who are admirable researchers and, more importantly, good people. Further thanks to those members who volunteered for my many experiments, probably boring for them, that were necessary to create my corpus. Special thanks to Dave Evans who has been a great office mate and friend; I sensed that we had a tremendous amount in common from our very first conversation (involving Hoagie Haven and palindromes), and that feeling has strengthened over the years. Thanks also to Noemie and Smara, who allowed me to interrupt them during so many of my breaks, even though they don't come to the gym. More thanks to Noemie for marrying one of my brother's best friends, which pretty much ensures she will always be around. Thanks to Elena for not minding too much that Dave and I like to eat in the office.

Thanks to all of my New York friends for making these the best years of my life so far to the point that I will not even remotely consider leaving this city. The work would have been so much harder if I were not happy in general, and I am happy because of my friends. Thanks to Fatima, Yaniv, Alex, Steve, Andy (who just organized Cardmania 2002), Jay (the founder of Cinematology), Neta, Sigalit, Susan, Alicia, Eleazar (for his positive influence), Stricker, Eric, and all those who will cause me regret when I realize later that I forgot to mention them. And, crazy as it may be, I thank the city for being such an amazing place to live. Thanks, of course, to my brother Evan for always being a super brother, and for making the same great choice as me (perhaps because we share a brain) by deciding to live in New York. Thanks to Veronica for marrying Evan and making him happy.

Thanks to my influences from earlier days. Thanks to Dave Kroll and Monib. We pushed each other to be better students in high school and college, we worked as a team during all of those unforgettable high school competitions, and we helped each other enjoy the very subjects that so many dread. Without having friends like these during those crucial years, who knows what path my education would have taken? Thanks also to the amazing teachers Mr. Papoula, Mrs. Stein, and Mr. Mezzadri, who increased my love of math and computer science, and who have certainly had a profound effect on my life. And finally, thanks to my parents, to whom this thesis is dedicated. They were not thrilled, at the time, with my decision to quit a job and become a graduate student, but I suspect they now realize it was the best decision I ever made. I am clearly the way I am because of them, and it is the very values with which they raised me that made the choice inevitable.

To my parents,

who have taught me to think logically and rationally

and to value education

Chapter 1

Introduction

1.1 Background

Text categorization, defined formally in Section 2.1, roughly refers to the automatic labeling of documents, based on natural language text contained in or associated with each document, into one or more pre-defined categories. Examples of text categorization tasks include the labeling of news articles into topical sections such as *Politics*, *Sports*, and *Entertainment*; the labeling of e-mail as *spam* or *not spam*; and the labeling of images based on captions as *Indoor* or *Outdoor*. Many additional pragmatic examples are discussed in Section 2.2. In recent years, due to the explosion of available, unlabeled information (e.g. on the Web or in digital libraries), there has been renewed interest in this area of research. Many advanced techniques have been developed (e.g. Support Vector Machines, Maximum Entropy) that seem to outperform older standards (e.g. Rocchio/TF*IDF, K-Nearest Neighbor, Naive Bayes), at least when applied to common text categorization collections. Still, there is no clear winner for all tasks, and research in the area is growing.

Almost all published research on text categorization deals with labeling textual documents, and in fact many authors (incorrectly, I would say) define text categorization as exclusively dealing with this task. However, I have pointed out in my research that the same techniques can be applied to non-textual documents, e.g. multimedia documents with associated text. In other words, using the text associated with a multimedia document, such as the caption of an image or perhaps closed-captions of a video segment, text categorization techniques can be applied to the text to automatically label the multimedia component. My research has thus concentrated on the automatic categorization of images based on associated text using (primarily) text categorization techniques. This task is important for many of the same reasons that it is important to categorize text-only documents. The increasing availability of free-floating images on the Web, the creation of large corpora of images, and the commonality of personal collections of digital photographs (some of which have annotations) all lead to the necessity of better ways to automatically label images to aid tasks such as browsing, filtering, and searching.

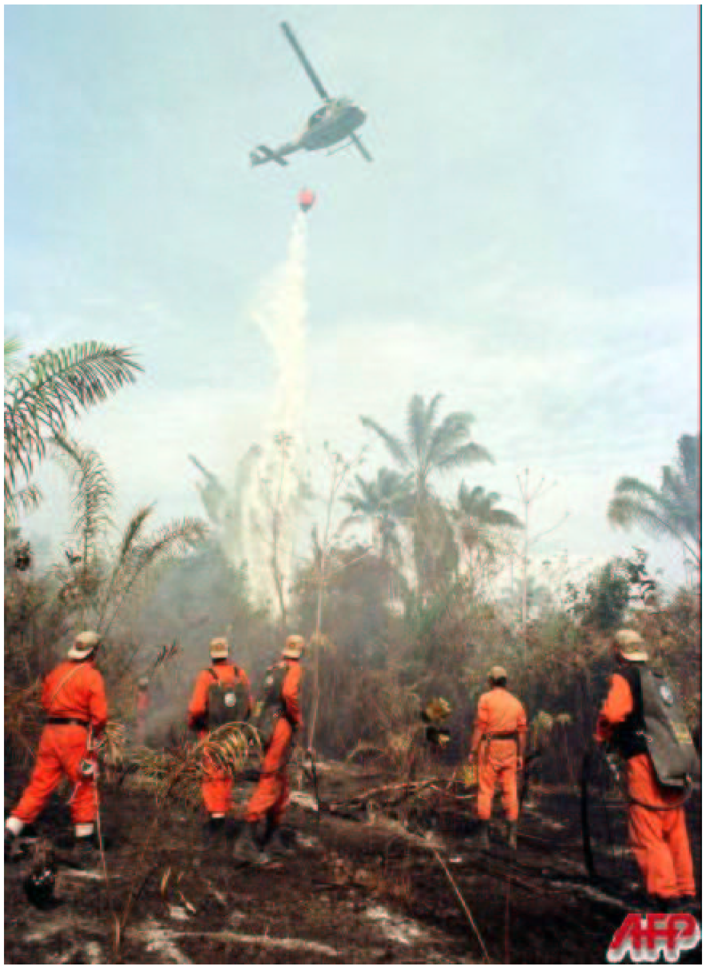
At times, standard text categorization techniques, without modification, apply well to the text associated with images, and part of this thesis describes two systems I have implemented, relying on approaches that are novel but still fall into the standard framework that is dominant in the field. However, the properties of the text associated with images, as well as the properties of the categories that are likely to be associated with images, are often quite different than those associated with full-length text documents such as articles, e-mails, or web pages. In addition, dealing with images gives access to a type of information that has not been available for text categorization research in the past; namely, the low-level *image features* of the image itself. In this dissertation, I show that these differences have opened the door for success in areas of research that have previously had mixed results, at best. More specifically, this thesis describes the positive results that I have achieved by combining systems and using deeper natural language processing (NLP) techniques to improve text categorization performance for certain tasks.

1.2 A Peek Ahead

Most of the experiments discussed in this thesis use a corpus that I created consisting of news documents with embedded captioned images; the creation of this corpus is discussed in detail in Section 3.1. Figure 1.1 shows a sample image from the corpus with its caption. Captions typically consist of two or three sentences, such that the first sentence describes the image and the rest gives background information about the related story. The entire document that the sample image comes from, including the associated article, is shown in Appendix A. This particular image is a member of three data sets. (A fourth data set associated with the corpus does not apply to this particular image.) One data set that applies to this image refers to the *Indoor* or *Outdoor* setting of the image, and this example is clearly an *Outdoor* image. Another data set that applies to the full document associated with this image refers to the type of event depicted; in this case, we are dealing with a *Disaster*. The third data set that applies to this image involves images embedded in *Disaster* documents and refers to the focus of the image; the category for this particular image is *Workers Responding*.

Just about all standard text categorization systems use a *bag of words* approach to represent documents. I explain this in great detail in Section 2.4, but basically this means that documents are represented as weighted vectors of the words appearing in the documents. Weights are generally computed by combining statistical features in some manner. Such systems do not rely on syntax or semantics when making decisions, nor do they use any additional information associated with a document.

For the task involving the *Indoor* versus *Outdoor* categorization of an image, and for the task involving the prediction of the type of event discussed in a news document, these approaches tend to work well. This is not all that surprising; if you look at the words in the caption of the sample image, you can see that several



APIAU, BRAZIL, 24-MAR-1998: Cordoba, Argentina, firefighters wait for a helicopter to drop water on a forest fire March 24 near Apiau, 120kms (74 miles) south of Boa Vista, capital of Roraima state. Some 120 Argentine firefighters and four helicopters are helping Brazilian counterparts battle the fires which began at the start of March and have been fed by the worst drought since 1926, which has been attributed to the El Nino weather phenomena. [Photo by Marie Hippenmeyer, AFP]

Figure 1.1: This is a sample image with its caption from the corpus used for most of the experiments discussed in this thesis.

are indicative of the correct categories. (When predicting the *Indoor* or *Outdoor* category of an image, it turns out that the first sentences of the caption is generally the most useful; when predicting the type of event of a news document, it turns out the full article is usually the most important.) The best systems, including two that I have designed and implemented myself, achieve an accuracy of over 85% for the *Indoor* versus *Outdoor* task and as high as 90% for the task involving types of events. Further improvement can still be achieved for the *Indoor* versus *Outdoor* task by taking additional information into account; namely, low-level image features such as color, something that is typically not available for text categorization tasks.

The task involving the prediction of the focus of images contained in *Disaster* documents defies standard text categorization systems. The same is true for a similar task involving the focus of images contained in documents about *Politics*. It turns out that, for these tasks, most of the words associated with the majority of images are not important, and some can even be misleading. The focus of an image, and the primary action taking place, are often indicated specifically by the main subject and the main verb of the first sentence of the image's caption. To make optimal use of this fact, I have implemented a system that relies on linguistic processing and a measure of word-to-word similarity. This system outperforms all standard systems tested for the task involving the *Disaster* image data set by a considerable margin, and it performs reasonably well for the *Politics* image data set. Further improvement can be obtained by combining very accurate rules, each of which only applies to a small number of cases, with the NLP based system or with standard systems.

Towards the end of the thesis, I discuss my work on Columbia University's Newsblaster system. This system does not involve the text categorization corpus I have created. Rather, it involves news documents (including embedded images) that are discovered and downloaded every day from many on-line news sources such as CNN, Reuters, and Fox News. Newsblaster is a system that showcases not only my own work, but also the work of many other members of Columbia's NLP research group. The system automatically collects, clusters, categorizes, and summarizes news from the web, and it provides a user-friendly interface to browse the results. Newsblaster utilizes my own research in two ways; news stories are automatically categorized into sections similar to those in newspapers and other news websites, and images are categorized with labels that help to provide users efficient browsing. Newsblaster has already caught the attention of the press and public. Articles about Newsblaster have already appeared in sources including The New York Times, USA Today, and Slashdot, and a recent analysis indicates that Newsblaster receives tens of thousands of hits a day.

1.3 Two Paradigms of Research

Most text categorization research fits into at least one of two paradigms. The first paradigm, which accounts for a vast majority of the recent text categorization literature, is research in machine learning, a concept that is explained in detail in Section 2.5. Such research generally deals with the exploration of new machine learning techniques and methodologies that seem to outperform existing techniques, at least for specific tasks or domains of interest. Some of the research I discuss in this thesis falls into this paradigm. For example, I have created two novel systems relying on techniques that, to the best of my knowledge, have not previously been applied to text categorization. One system uses a statistical technique known as *density estimation* to refine the output of standard systems and to provide probabilistic confidence measures for predictions. The other system uses an approach known as *binning* to empirically estimate term weights for groups of words that share features in common, thus avoiding inaccurate term weights for individual words with scarce evidence. Other research I discuss in this thesis that fits into the first paradigm involves the combination of a set of automatically discovered rules, each of which is very accurate but rarely applicable, with other systems to improve performance.

The second paradigm of research deals with representation; in other words, the exploration of which features of a document are important. This type of research has been far less common in the recent text categorization literature. (Exceptions are discussed in Section 2.4.3.) Almost all text categorization systems today use a bag of word representation for documents, and although there are several choices to be made even within this way of thinking, the exploration of representation rarely deviates substantially from the norm. However, the focus of my research on the categorization of images leads to important differences from the categorization of lengthy text documents such as articles, e-mails, or web pages. I show in this thesis that the properties of the text and categories that are often associated with images sometimes necessitate the need for the consideration of syntactic features; furthermore, the availability of low-level image features can, at times, provide additional clues for an image's category. My work considering the use of deeper NLP techniques to improve categorization, as well as my work considering the use of low-level image features, falls into the paradigm of using novel representation for documents.

Machine learning approaches to categorization are general. They may not achieve the best possible performance on any given task, but they can usually be applied to many similar tasks. Once a machine learning system has been developed, it can be applied to new tasks by simply giving it training data (i.e. examples of what it is supposed to learn). Research dealing with novel representation is often more specific. For example, in Section 2.4.3, I discuss research involving the use of link structure to improve retrieval from the World Wide Web, and clearly this only applies to web pages. My own work involving a combination of text and low-level image features obviously applies only to images. Although such research may not be as general as other machine learning research, it is often very interesting and sometimes leads to considerable performance gains over standard approaches.

1.4 Contributions

This thesis includes contributions that fall into both research paradigms discussed in Section 1.3, and also contributions that are more general (e.g. plugging my research into Columbia's Newsblaster system). Several of the contributions deal with the use of robust statistical techniques to categorize images. A few concern the exploration of hybrids that combine different approaches or integrate different types of information (e.g. NLP and traditional IR; high-precision, low-recall rules and other systems; text and image features). The major contributions of my thesis are as follows:

- *The exploration of the use of text to categorize images.* In order to have the means of conducting such research, I have created a multimedia corpus consisting of news documents with embedded captioned images and multiple data sets representing various levels of abstraction.[1]

- *The introduction of two novel machine learning approaches towards text categorization involving the use of density estimation and bins.* Density estimation provides a way of estimating a probabilistic confidence measure for each of a system's predictions, and it often also improves accuracy. The use of bins provides a mechanism for empirically estimating accurate term weights for words with scarce evidence, and for determining which statistical features

[1]This corpus, described in detail in Section 3.1, is ready to be made publicly available for future researchers. Once this happens, I will post instructions describing how to obtain the corpus at http://www.cs.columbia.edu/~sable/research/corpus.html.

of a word are important.[2]

- *The integration of NLP techniques and traditional IR techniques for categorization.* I show that for certain categories referring to the focus of an image, deeper linguistic processing is necessary for optimal performance.

- *The use of high-precision, low-recall rules to improve the results of other systems.* I demonstrate how to improve accuracy for tough text categorization tasks by combining rules, each of which is rarely applicable but very accurate when it applies, with systems to which we can fall back when the rules do not apply.

- *Combining text and image features for the categorization of images.* Using low-level image features is necessary when no text is available, and when both are available a combination of the two can perform better than either individually.

- *The categorization of news and an image browsing interface for Newsblaster.* I discuss my personal contributions to this popular system that automatically extracts, clusters, categorizes, and summarizes news and related images from the web. Newsblaster receives tens of thousands of hits every day and has been discussed in sources including The New York Times, USA Today, and Slashdot.

The first bullet is the core of this thesis, and the other contributions all relate to this in some way. The new approaches can be applied to text associated with images directly, and for certain data sets they achieve excellent results without the use of more unusual techniques. The third through fifth bullets represent lines

[2]The BINS system, described in detail in Chapter 5, is ready to be made publicly available for research purposes. Once this happens, I will post instructions describing how to obtain the system at http://www.cs.columbia.edu/~sable/bins.html.

of research that researchers have been experimenting with for decades with results that have been mixed, at best; however, both the categories and the text associated with images differ substantially from those used for text categorization of full-length textual documents, and for these reasons I have found that the integration and combination of various approaches can be helpful. The final bullet represents work that demonstrates the pragmatic benefits of this thesis in a way that has already grabbed the attention of the public and press.

1.5 Overview of Dissertation

Chapter 2 provides a survey of text categorization techniques and related information necessary to understand and appreciate the rest of this dissertation. Included in this chapter are detailed discussions of bag of words representations for documents, general machine learning concepts, common approaches towards text categorization, common evaluation metrics, etc. I also argue in this chapter that the reason that there are so many text categorization techniques with no clear winner for all text categorization tasks is analogous to the "No Free Lunch" theorems.

The next few chapters represent novel work that still fits into the general framework that is common in the text categorization literature; in other words, these chapters deal with the application of existing techniques to new domains and also with the creation of new machine learning techniques. Chapter 3 represents the core of this dissertation; here, I discuss the categorization of images based on associated text. I also describe, in this chapter, the creation of a corpus consisting of news documents with embedded images and captions, for which I have defined several sets of applicable categories and collected manual labels. Chapters 4 and 5 present my own contributions to the general text categorization literature; here, I discuss the use of density estimation and bins to improve standard text categorization techniques. Density estimation provides a way of estimating a probabilistic confidence measure for each of a system's predictions, and it often also improves accuracy. The use of bins provides a mechanism for empirically estimating accurate term weights for words with scarce evidence, and for determining which statistical features of a word are important. For some image categorization tasks, these systems work very well without modification.

The next few chapters deal with research that is more unusual in the text categorization literature. These chapters concern issues of representation of documents and combinations of systems or approaches; these are lines of research that generally have not proven helpful for text categorization in the past, but have been useful for the categorization of images. More specifically, Chapter 6 discusses the use of deeper linguistic processing and a novel measure of word-to-word similarity to improve performance for a specific text categorization task. Chapter 7 involves the combination of very accurate rules that do not apply very often with other systems to improve performance for difficult tasks. Chapter 8 involves image categorization using low-level image features, either by themselves or in combination with text.

Chapter 9 discusses my work on Newsblaster, a system that showcases NLP research including text categorization of both images and articles. Newsblaster, which has already captured the attention of the public and the press, is a publicly accessible web-based system that automatically extracts, clusters, categorizes, and summarizes news and related images from the web. Updated daily, this system demonstrates some of the the pragmatic benefits of my research by providing users a useful and interesting way to access current news. Finally, Chapter 10 summarizes the main contributions of this dissertation and discusses potential future work.

Chapter 2

Text Categorization Survey

This chapter presents a survey of text categorization, including descriptions of techniques, evaluation measures, common corpora, etc. This chapter can serve as a standalone tutorial for anyone interested in learning about this area of research. A basic understanding of this field is necessary to place the research discussed in the rest of the thesis in its appropriate context. At the end of the chapter (Section 2.9), I discuss some of the ways that properties of documents and categories involved with text categorization tasks can vary; I then argue that it is for a reason analogous to the "No Free Lunch" theorems that there is no clear winner out of all the existing techniques for text categorization tasks. *For readers of this thesis who are already familiar with the field of text categorization and are more interested in my own personal contributions, this chapter may be skipped.* I refer to appropriate sections of this chapter throughout the rest of the thesis, so readers who encounter unfamiliar material can look back and review specific sections of the chapter when necessary.

Many of the descriptions in this chapter and throughout the rest of the thesis involve mathematical equations or descriptions. To avoid confusion, I have tried to be consistent with my notation for various concepts that occur frequently. The notation that I use for some of these common concepts is summarized in Table 2.1.

Expression	Meaning
D	A set of documents
d	A single document (typically $d \in D$)
C	A set of categories
c	A single category (typically $c \in C$)
T	A set of terms (which may or may not be words)
t	A single term (typically $t \in T$)
W	A set of words
w	A single word (typically $w \in W$)
N	The number of documents in a set (typically $N = \|D\|$)
$DF(w)$	Document frequency of w (the number of documents in which w occurs at least once)
$TF(w)$	Term frequency of w in a set of documents (the number of occurrences of w in a set of documents)
$TF(w, d)$	Term frequency of w in a single document d (the number of occurrences of w in d)
$IDF(w)$	Inverse document frequency of w, a measure of rarity ($IDF(w) = -\log_2[DF(w)/N] = \log_2[N/DF(w)]$)
λ	A weight
I	An indicator function (0 or 1 depending on some condition)

Table 2.1: The reader can refer back to this table that lists the mathematical notation commonly used throughout the thesis.

Of course, from time to time there are various other concepts that come up for which I introduce new variables, and I describe these whenever this happens. Common algebraic variables such as i, k, n, and x are used to represent different values in different places, so the meanings of these change throughout the thesis.

2.1 Definition of Text Categorization

As described at the start of Chapter 1, text categorization refers to the automatic labeling of documents, based on natural language text contained in or associated with each document, into one or more pre-defined categories. Some text categorization tasks assume the categories are independent, in which case each document can be assigned to no categories, one category, or multiple categories. In such cases, the categories are said to be binary categories, and a separate YES/NO decision is required for each category/document pair. More formally, let D be the set of all documents and C be the set of all binary categories; a text categorization system then has to assign a Boolean value to each $(d,\ c)$ pair, where $d\ \in\ D$ and $c\ \in\ C$. Other text categorization tasks assume that the categories are mutually exclusive and exhaustive, in which case each document is assigned to exactly one category. Then the inclusion of a document in one category excludes the inclusion of the document in all other categories. In this case, the classifier must map each document $d\ \in\ D$ to the category $c\ \in\ C$ that is the best fit.

It should be noted that certain tasks can be viewed as fitting into either of the two paradigms described in the previous paragraph. Text categorization tasks with exactly two mutually exclusive categories can be viewed as a binary categorization task involving either one of the categories, such that any document that doesn't belong to this category is automatically placed in the other category. For example, later in this thesis, I discuss the categorization of images as either *Indoor* or *Outdoor*; I could, instead, view the decision being made as a binary decision as to whether or not the image is *Indoor*. In this particular case, however, *Indoor* and *Outdoor* have equally relevant semantic meaning, and viewing them as two mutually exclusive categories as opposed to one binary category is - at least in my mind - more intuitive. On the other hand, if the task is to categorize websites as *pornography* or *not pornography*, the *not pornography* category is vague, consisting of all other types of websites, and it seems more natural to view this as a binary decision as to whether or not the website is pornographic. Then there are cases that fall somewhere in between; for example, categorizing movie reviews as positive or negative can easily be thought of fitting into either paradigm. In any case, either type of text categorization system (i.e. those designed to work with mutually exclusive categories or those designed to work with binary categories) can be applied to these tasks.

Another special case of text categorization tasks occurs when mutually exclusive categories form a hierarchy, also known as a taxonomy.[1] One commonly known taxonomy is the Yahoo directory of web pages set up for easy browsing.[2] Some systems take such hierarchical structures into account by using a ladder approach. Instead of considering the bottom nodes of the hierarchical tree directly, a decision is first made at the top node of the tree as to which of its immediate children is appropriate, then at the second level, etc., until a terminal node is reached. The idea behind this is that each decision being made is simpler than an immediate decision involving all possible categories, especially when the taxonomy is large. Some researchers have found noteworthy improvement using this approach (Koller and Sahami, 1997; Chakrabarti et al., 1998; Ruiz and Srinivasan, 1999; Dumais and Chen, 2000; Weigend, Wiener, and Pedersen, 1999). Other research involving text categorization with hierarchical categories includes that of (McCallum et al., 1998) in which the accuracy for small categories with sparse evidence is improved using a smoothing technique that considers these categories' parents. The corpus I have personally created, described in Section 3.1, contains hierarchical categories, and although I have not exploited the hierarchical nature of the categories myself, I hope that the corpus becomes a popular resource for text categorization researchers, and this is clearly an issue that merits further exploration.

This thesis and the survey of the field presented in this chapter concentrate on tasks involving mutually exclusive categories and techniques that do not consider hierarchical structures. At times, I discuss binary categorization tasks and related

[1]Technically, a hierarchy could be any directed, acyclic graph, but it is common to only consider trees (Chakrabarti et al., 1998).

[2]see http://www.yahoo.com.

issues, but my focus is on tasks involving mutually exclusive categories, which I feel are underrepresented in the text categorization literature. The major reason for this underrepresentation is probably that the Reuters corpus, the most common publicly available text categorization corpus (described in Section 2.8) consists of binary categories. I have found in my own research, however, that many (perhaps the majority) of text categorization tasks that arrive naturally involve mutually exclusive categories. See, for example, the list of potential applications for text categorization provided in Section 2.2, and you will note that many of them fit this paradigm. For a survey of text categorization that focuses on binary text categorization tasks, I recommend a recent work by Sebastiani (Sebastiani, 2002).

2.2 Applications of Text Categorization

There are a wide variety of useful applications for text categorization. Work dating back to the early 60's uses text categorization techniques for the purpose of authorship attribution; for example, Mosteller and Wallace (1963; 1964) have addressed a long standing dispute as to which of two authors, Alexander Hamilton or James Madison, was responsible for a part of a collection of anonymously published articles jointly known as The Federalist papers. Some of the more current applications of text categorization include: the classification of e-mail as *spam* or *not spam* for filtering purposes (Sahami et al., 1998; Drucker, Wu, and Vapnik, 1999); the classification of news articles into topical sections (e.g. *Politics*, *Sports*, *Entertainment*, etc.) for browsing (Li and Jain, 1998; Sable and Church, 2001; McKeown et al., 2002); classification of websites as *pornography* or *not pornography* for filtering purposes (Lewis and Schutze, 2002); hierarchical classification of websites into a large variety of topics for browsing (McCallum et al., 1998; Chakrabarti et al., 1998; Dumais and Chen, 2000); extraction of certain types of metadata from text documents or websites to improve search capabilities (Chakrabarti, Domb, and Indyk, 1998); the classification of reviews (e.g. reviews of movies or travel destinations) as *positive* or *negative* to summarize statistics (Pang, Lee, and Vaithyanathan, 2002; Turney, 2002)[3]; the automatic grading of essays, with the categories being the possible grades, to minimize human workload (Larkey, 1998); etc.

Many other common classification tasks could potentially be viewed as text categorization tasks. For example, word sense disambiguation refers to the automatic classification of words in a document with their correct, current sense. This can be viewed as a text categorization task where the categories would be the possible senses, which would be different for each possible word, and the text used for the categorization would be the word's context (i.e. the text surrounding the word) (Gale, Church, and Yarowsky, 1993). Information retrieval (van Rijsbergen, 1979) refers to the automatic discovery of documents in a corpus that match a user's query. This can be thought of as a text categorization task where the categories are *relevant* and *not relevant* (but the meanings of these categories would change with each new query). Topic detection and tracking (TDT) (Wayne, 2000) refers to the detection of new stories that are similar to a set of example stories. This can be viewed as a text categorization task where the categories are *similar* and *not similar* (in this case, the meanings of these categories change for each set of stories). This thesis concentrates on tasks like those in the previous paragraph, for which there is a single set of categories that apply to all instances, but many of the same techniques can be used to the tasks mentioned in this paragraph as well.

2.3 Other Tasks Related to Text Categorization

The Natural Language Processing (NLP) literature is filled with terms that are similar to, or related to, *text categorization*. Unfortunately, there is not complete

[3]If this were accurate enough it could save many human hours for the people who work for sites such as http://www.rottentomatoes.com, one of my personal favorites on the web.

uniformity among authors, and some terms are used in different ways depending on the source. In this section, I discuss some of these terms. These are the ones that I consider to be the most important, *and I am defining them in what I consider to be the most common, useful, and/or intuitive manner*. This discussion is meant to avoid confusion when these terms show up in the rest of this thesis or elsewhere. At the end of the section, I discuss the ambiguity of some of the terms.

Clustering (Rasmussen, 1992) refers to the automatic grouping of text documents such that all documents within a single group are similar to each other as determined by some statistical measure. For example, I show in Chapter 9 that the Newsblaster system developed at Columbia University uses clustering to create groups of news articles such that each group consists of articles describing the same event. Unlike text categorization, when dealing with clustering there are no categories defined in advance. There is no pre-defined concept that any given cluster represents; each cluster simply consists of documents that are similar to each other.

Information retrieval (IR) (van Rijsbergen, 1979) refers to the automatic discovery of documents in a corpus that match a user's query. While this can be considered a special instance of text categorization in which the categories are *relevant* and *not relevant*, the actual meanings of these categories change for every query. One very common use of information retrieval is searching the World Wide Web with a search engine such as Google.[4] Some of the same techniques that are useful for text categorization are also useful for information retrieval, but there are important distinctions, due to the existence of user queries, that lead to divergence between the two fields.

Topic detection and tracking (TDT) (Wayne, 2000) deals with finding a set of stories related to a set of example stories. For example, users may be interested in reading news articles concerning specific events. This can be viewed as a special

[4]available at http://www.google.com.

instance of text categorization in which the categories are *similar* and *not similar*, but the meanings of these categories change for every user. Typically, user profiles are set up containing examples of documents that users would want (or perhaps not want) to see. These user files may be learned based on users' actions, or they might be based on relevance judgments of documents by users. In practice, the techniques used to perform topic tracking are more likely to come from the field of information retrieval than from the field of text categorization.

Text filtering (Schapire, Singer, and Singhal, 1998), sometimes called *information filtering* (Belkin and Croft, 1992), refers to the streaming of incoming information such that a user is only presented with new documents that he or she would likely want to see. Examples include filtering e-mail to remove spam or pornography. Text filtering is a specific case of binary text categorization in which the categories are *relevant* versus *not relevant* (or *interesting* versus *not interesting*, or *useful* versus *not useful*). Unlike information retrieval and topic tracking, the meanings of these categories do not change based on queries or user profiles, and so text filtering, the way I am defining it here, is clearly a subset of binary text categorization. There are, however, certain distinguishing characteristics of this task. For example, since text filtering usually deals with documents that arrive over time (e.g. e-mails), certain methods that require the entire test corpus to be available at one time do not apply.

Text classification is commonly used as a more general term that can refer to any of the topics previously described in this section. However, it should be noted again that there is not consistency among authors. In some works, for example, authors use the term *text classification* in a more restrictive sense, referring to the topic that I call *text categorization.* In the other direction, I have seen several cases of *text categorization* used in the more general sense (as I use *text classification*), and I have also seen it used to refer to a combination of the two areas that I refer to as *text categorization* and *clustering* but not *information retrieval.* In other cases, I have seen *information retrieval* used in the general sense, and I even catch myself using it in the general sense when I refer to the existing *information retrieval literature*; however, due to the word “retrieval”, I think it is best used to refer to query/retrieval applications, as it is often used in the literature. I have also seen *text filtering* used in a more general sense to include the topic that I call *topic tracking.* In any case, it is probably not so important to remember all of the tasks to which, at times, each term applies; it is usually clear enough from context, so as long as the reader realizes that there is not complete uniformity among authors, that should be enough to avoid confusion.

2.4 Representing Documents Using Bag of Words Approaches

Text categorization refers to the automatic labeling of documents based on text. In order to label documents, systems must first be given access to each document, and the document must be *represented* by the system in some way. Almost all modern text categorization systems represent documents using what is known as *bag of words* approaches. This means that each document is represented as a vector of weighted words, although exactly what constitutes a word can vary to some degree as is discussed later in this section. Weights are generally computed by combining statistical features of words in some manner, as is also discussed in this section.

Let d be a document that needs representation, and T be the set of terms used by the representation. If a bag of words approach is being used, each term is a single word, and T is the set of all possible words. Let λ_t be the weight of a single term $t \in T$. Then d can be represented as $d = [\lambda_{t_1}, \lambda_{t_2}, ..., \lambda_{t_{|T|}}]$. Bag of words approaches do not rely on syntax or semantics; in other words, the ordering of terms in a document does not matter, and the relationship between any two distinct terms is not considered. Although there are many possible weighting schemes, some of which are discussed in Section 2.4.2, one thing that most have in common is that words that do not appear in a document are assigned a weight of zero. This turns out to be helpful, since most documents likely do not contain most of the possible words, and it is then only necessary to keep track of weights for words that do occur in each document.

2.4.1 Words as Terms

By far, the majority of text categorization systems that I am aware of use single words as terms when representing documents. It is generally accepted in the information retrieval and text categorization literature that more complex representations do not lead to improved performance and are often less efficient (Salton and Buckley, 1988; Lewis, 1992; Sebastiani, 2002). It might seem that using phrases such as bigrams (instances of two consecutive words) or trigrams (instances of three consecutive words) instead of, or in addition to, single words might be useful, since some context would be accounted for; however, for most bigrams and trigrams, there is scarce evidence, since the number of existing instances of bigrams and trigrams in a corpus is miniscule compared to the possible number of bigrams and trigrams in a language (the square and cube of the number of words in the language, respectively).

Although the approaches I am discussing use single words as terms, there are still several decisions to be made as to what exactly constitutes a word, and how to distinguish words automatically. Most systems do not have a complete dictionary of allowable words, and so they must use general rules to distinguish words from non-words. One possible simple rule (actually used by my BINS system described in Chapter 5) is to consider a word to be any sequence of alphabetic characters, with any other character considered a word separator. This has an unusual effect when apostrophes are used in an actual word (e.g. “it's” is considered to be two words, “it” and “s”), but I have found the the effects of such cases are minimal, and performance is not improved by more complex methods. (For example, an earlier version of my BINS system, and also my density estimation system described in Chapter 4, use Church's statistical part-of-speech tagger, POS (Church, 1988), to determine word boundaries. When I updated BINS to the simple rule, performance did not degrade.)

Even after the general rule for determining word boundaries is decided, there are still many options that systems have in determining which, if any, transformations must apply to each word. For example, case sensitivity determines whether or not two words that differ only in terms of capitalization should be treated as equal. In other words, if two words are the same except that one has one or more letters capitalized whereas the other has the same letters in lower case, should they be considered two instances of the same word or two separate words? At times, capitalization can mean the difference between a common word or a proper noun, but at other times, it can be the result of placement at the beginning of a sentence.

Another issue is stemming, a simple rule-based technique that converts different forms of a word to the same root token, or stem. For example, different tenses of a verb such as “help”, “helps”, “helped”, or “helping” are all converted to the same stem “help”, whereas different forms of a noun such as “house” or “houses” are all converted to the same stem “hous”. However, common stemming algorithms (e.g. Porter stemming (Porter, 1980)) do not handle irregular verbs or unusual nouns correctly. At times, different forms of the same root fail to be converted to the same token (e.g. “child” stays as “child” and “children” stays as “children”), while, at times, forms of different words do get converted to the same token (e.g. “tire” and “tired” both get converted to “tire”). More accurate than stemming is to use a lexical database such as WordNet (Fellbaum, 1998) to convert every word to its morphological base-word. However, this is also much more time consuming than stemming, and it is still not guaranteed to improve performance, as for certain text categorization tasks, the specific tense or form of a verb or noun may give a clue as to the appropriate category (for example, see (Riloff, 1995), in which distinctions between singular and plural nouns or passive versus active tenses for verbs made a significant difference for her categorization task).

Another issue is whether to use all words for text categorization or to filter some out. Many (I believe most) current systems use a stop-word list, a hard coded list of common words such as articles and prepositions that are ignored when they occur. Although these words would be given very low weights by machine learning methods (some are described later in this chapter), there tend to be many of them in a document, and so the noise introduced by these words can add up and make results less accurate. Another possibility is to automatically determine which words to filter out based on weights; for example, the next subsection of this chapter describes the inverse document frequency (IDF) weighting scheme, and this can be used to exclude all words with IDF values under some specific constant (this is explored in (Sable and Hatzivassiloglou, 2000) and also Appendix G). However, one must be careful when deciding whether or not to filter out such words, as they can be helpful for certain text categorization tasks. An excellent example is the breakthrough work of Mosteller and Wallace, the first application mentioned in Section 2.2, in which the authors determine that filler words such as “an”, “of”, and “upon” are very important for an authorship attribution task, whereas more meaningful content words are not. Another example is the work of Riloff (1995), in which the author has found that small words can combine with more important words to form “relevancy signatures” that are very indicative of categories.

Even when two instances of two words are spelled the same way with the same case sensitivity, they may not really represent the same word with the same meaning. For example, “can” is sometimes a noun (as in “I ate a *can* of beans.”) and sometimes a modal verb (as in “You *can* do it!”); “wind” is sometimes a noun (as in “There’s a strong *wind* out today.”) and sometimes a verb (as in “I need to *wind* my watch.”). Such pairs of distinct words that are spelled the same are known as homographs, and ideally, a system should probably distinguish one sense from another, although few text categorization systems even try. One way that systems could differentiate homographs of each other when the two words represent different parts is to use a part-of-speech tagger (such as POS, mentioned earlier in this section); the part-of-speech tag assigned to each word by the tagger can be included as part of the token (this is explored in (Sable and Hatzivassiloglou, 2000) and also Appendix G).

In my own experience, I have found that most of these issues involving what constitutes an individual word are not particularly important for text categorization; i.e. performance is not significantly affected regardless of the decisions made. An exception is this last issue (distinguishing homographs using part-of-speech tags) which seems to be somewhat helpful, although this requires a tagger and more time to process texts. Some of these features are examined in more detail in Appendix G, in which cross validation experiments (to be explained in Section 2.5) are used to determine the optimal values of features for a system I have developed. Although most choices involving the precise method to distinguish words seem to matter to only a small agree, other features explored in that appendix (e.g. what text span to consider in a document) prove to be quite important.

2.4.2 Common Weighting Schemes for Words

Once all of the decisions discussed in the previous subsection have been decided, a system can process a text document and determine which words are present. At that point, there are many possible weighting schemes that can be used to set up the vector representing the document. The simplest possible measure is a binary value for each word; 1 if it is present and 0 otherwise. In this case, the bag of words approach is also known as a *set of words* approach.

More typically, systems use more complex weights. One approach that is still somewhat simple is to weight each word in a document with the word's *term frequency* (TF) (Salton and Buckley, 1988; Salton, 1989), which is the number of times that the word appears in the document. All other things being equal, it is expected that words that appear more often in a document are more important as to the overall meaning of the document. Consider, for example, a document that mentions Afghanistan once, compared to a document of equal length that mentions Afghanistan ten times; you would probably expect the second document to have more of a focus on that topic.

Of course, words such as "the" will likely appear many times in a document without being considered important for most text categorization tasks. Usually, a system combines term frequency with some other measure to take this into account. The most common measure that is often combined with term frequency is *inverse document frequency* (IDF) (Salton and Buckley, 1988; Salton, 1989), which can be thought of as a measure of a word's rarity in a corpus (or a language, but this needs to be estimated based on a corpus). It is believed that words that are rare tend to be more specific, and may therefore contribute more to content. Let $DF(w)$ be a word's *document frequency* (DF), which is the number documents out of a set of N total documents that contain one or more instances of the word. Then the IDF of the word can be computed as follows:

$$IDF(w) = -\log_2 \frac{DF(w)}{N} = \log_2 \frac{N}{DF(w)} \tag{2.1}$$

Therefore, very common words that appear in almost all documents have an IDF close to zero, whereas rare words have larger IDFs, with a maximum possible value of $\log_2 N$ for a word that only occurs in one document. (Words that are never seen in the N documents have an undefined IDF and, thus, are generally ignored; this makes sense if the N documents are those of the training set, since there is then no evidence indicating one category versus another for these words.) The specific logarithmic base being used is actually not important so long as it is consistent.

The most common method of combining a word's term frequency with its inverse document frequency is to simply multiply the two weights together:

$$TF * IDF(w) = TF(w, d) \times IDF(w) \tag{2.2}$$

This TF*IDF representation effectively combines a word's importance to a document with its specificity over a corpus. At times, you might see more complex combinations of the same measures or similar measures, but ever since the very influential work of Salton and Buckley in the late 80's (Salton and Buckley, 1988; Salton, 1989), TF*IDF has dominated weighting schemes in the text categorization and information retrieval literature.

Typically, TF*IDF weights for words are computed based on the same data used to train a system (the concept of training is discussed in the next section). An alternative is to compute the values based on some other, larger corpus. On the one hand, there would be more data, since vast amounts of unlabeled data is easily obtainable, but on the other hand, the data would likely not be as representative of future documents. There is quality versus quantity issue at play. I discuss this in more detail in Appendix K. In summary, the results are inconclusive; using IDF weights from a larger, outside corpus sometimes improves performance and sometimes degrades performance. The appendix explains why I have chosen to use the training set by default for my BINS system (described in Chapter 5).

Many systems normalize the final vectors used to represent documents. Normalization prevents larger documents from being considered more similar to categories or other documents simply because term frequencies tend to be higher when documents are longer. For certain approaches that also use vector representations to represent centroids of categories (e.g. the Rocchio approach discussed in Section 2.6.1.1), it is more important to normalize the category vectors.

2.4.3 Exceptions to Bag of Words Approaches

One noteworthy exception to bag of words approaches is work involving latent semantic indexing (1990), in which documents are represented as combinations of abstract "concepts" determined by mathematical analysis using singular value decomposition (SVD). This technique helps avoid the problems of synonymy and polysemy by providing a mathematical way of determining the "similarity" between concepts. Another exception is the string kernel method proposed in (Lodhi et al., 2002), in which the terms are all subsequences of k characters that appear in a document (the authors consider multiple possible values of k). You will encounter another exception in this thesis in Chapter 6 (or in (Sable, McKeown, and Church, 2002)), which describes an NLP based system that uses shallow parsing to extract only the main subject and verb from the first sentences of image captions, and represents documents using only these two words. For certain, specific tasks, additional features of documents are sometimes taken into account along with words; for example, for the automatic grading of essays, systems may consider the length of a document or the average sentence length (Larkey, 1998).

Then there are methods that start with bag of words approaches but afterwards also consider additional information. In (Kleinberg, 1999), the author describes how to utilize link structure to find pages on the Web that are relevant to a query. The author defines the notion of "authorities" (respected sites, likely to have many other sites point to them) and "hubs" (pages that link to many authorities), and describes how to use these concepts to supplement traditional techniques utilizing text and bag of words approaches. A similar idea is discussed in (Brin and

Page, 1998), and the algorithm described here is used by the enormously popular and successful Google search engine. Also applying to web pages, the work discussed in (Chakrabarti, Domb, and Indyk, 1998) takes hyperlinks into account for a text categorization task.

For the categorization of images, systems may rely on a combination of bag of words approaches applied to image captions with approaches that rely on low-level image features. This is discussed in Chapter 8 of this thesis (and in (Paek et al., 1999)). In Section 8.4, I mention several research endeavors involving the categorization or retrieval of images based only on image features; this, of course, does not fall into the domain of text categorization at all, although sometimes similar machine learning techniques are used (machine learning is explained in Section 2.5, and some common methods are explained in Section 2.6).

The exceptions to bag of words approaches discussed in this subsection often apply to specific tasks (e.g. the automatic grading of essays) or specific types of documents (e.g. web pages or images). Although these research endeavors involving novel representation of documents are not always as general as other machine learning research, they are often very interesting, and they can lead to significant improvement over standard approaches. Still, the vast majority of the text categorization literature focuses on bag of words approaches, and the rest of this chapter, which is a general survey of the field, is thus limited to these; later in the thesis, research involving novel representation is discussed again.

2.5 Training, Testing, and Machine Learning

Almost all modern text categorization systems use machine learning techniques (Mitchell, 1997) to "learn" how to predict categories for documents.[5] In other

[5]One exception is an Expert System, which is basically a set of hard coded rules designed by an "expert" to determine a document's category. A major disadvantage of this approach is that

words, training data must be provided to the system before it can be applied to unlabeled documents. For text categorization tasks, this data comes in the form of manually labeled documents. For binary categorization tasks, this requires that there are positive and negative examples of each category, so that the system can determine one from the other. For tasks involving mutually exclusive categories, as focused on in this chapter and thesis, this requires that there are examples each category. How many examples are necessary for reasonable performance depends on the specific task and the method being used. For a given task, it is quite possible that the method that performs the best with a small amount of training data may not be the method that performs the best with a large amount of training data.

The advantage of a machine learning approach to automatic text categorization is that it is general. Once an implementation of any such method exists, all that is needed to move to a new set of categories is training examples. In fact, creating a corpus of such training examples is often the most time-consuming part of moving to a new set of categories. There are certain text categorization tasks for which the labels are obvious from the start - for example, determining what news group an article comes from - in which case this phase can be skipped; but for most text categorization tasks, automatic creation of a training set does not work well. See, for example, Appendix P, in which I discuss an attempt by researchers at Columbia to automatically create a training set for our Newsblaster system, described in Chapter 9, in order to avoid the need of manually labeling documents. I show in that appendix that this attempt did not succeeded, and that eventually it became necessary to hand label documents for training purposes. In this case, it turns out that the quality of the training data is more important than the quantity of the training data.

---

it is completely domain dependent, and new rules have to be generated for every set of categories, often involving separate experts. In addition, expert systems do not always perform as well as machine learning approaches, since the best rules for categories are not always intuitive.

A few issues that must be decided when collecting labels for documents are:

- How many documents should be used for training?
- How should labels for the documents be collected?
- How many volunteers should label each document?
- What level of agreement should be required in order to use a document?

Of course, as the amount of desired training data increases, and as the number of labels required for each document increases, the more person hours are necessary to actually do the labeling. In Chapter 3, I cover the decisions I made concerning these issues as part of a discussion of the creation of a corpus consisting of over 2,000 documents and four sets of categories. I have set up a user-friendly web interface that allows volunteers to suggest categories for documents. I have personally labeled all of the documents in every set of categories, and many volunteers have also labeled documents such that each document has received a label for each set of categories by one volunteer. If there was agreement between me and the volunteer for a particular set of categories, I would include the document in the final data set for that set of categories.

Once labeled data has been collected, it is often the case that systems using various approaches need to be evaluated. Sometimes, this is for research purposes, so that multiple approaches can be compared to each other. At other times, for pragmatic benefits, it is often desirable to determine which is the best of available systems to use for a specific task, and also to have an idea of expected future performance. The labeled corpus must therefore be divided into at least two sets, one consisting of data used to train the system (this portion of data is known as the *training set*), and another consisting of data used to test the system (this portion of data is known as the *test set*).[6]

[6]Why there is not consistency in this naming convention - i.e. why the sets of data are not

One common technique for dividing a corpus intro a training set and a test set is to decide how large one of the two sets needs to be and then randomly selecting this number of documents for this set, placing the remaining documents in the other set. This is a common approach, and one that I often take in my own work (e.g. see the descriptions of data sets I have defined for my own corpus as described in Section 3.1). However, there are certain cases for which this is not desirable. For example, in the news domain, the focus of news tends to change with time. If a corpus of labeled documents includes articles from a particular time frame, dividing this corpus randomly into training data and test data would likely include some articles concerning a particular story in the training set and other articles concerning the same story in the test set; for this reason, the test documents would probably be closer in content to the training documents than future documents will be. Thus, if one wants to obtain accurate performance estimates for future news documents, it is perhaps better to test using material that is from a different time frame than the training material. This is the approach I take in Appendix P, which discusses my evaluation of the performance of categorization for the Newsblaster system (described in Chapter 9); the data used for testing is one day of actual Newsblaster articles that is from a later time period than any of the training material.

Often, each system that is being evaluated has one or more optional parameters for which settings need to be chosen. This generally involves tuning the parameters by performing initial experiments that test out multiple (perhaps all) possible combinations of settings. Based on these initial experiments, a single set of settings is chosen for the system. Here I discuss two common methodologies for choosing appropriate settings for parameters.

One common methodology for choosing settings of parameters involves di-

known as a training set and a testing set, or a train set and a test set - is something for which I have never heard an explanation.

viding the labeled data into three sets instead of two; in addition to a training set and a test set, there is also a *tuning set* (also known as a *validation set* or a *hold-out set*) (Kohavi, 1995; Lam, Ruiz, and Srinivasan, 1999; Sebastiani, 2002). For each possible setting (or combination of settings), the system is trained on the training set and tested on the tuning set. Based on the results of these experiments, settings are chosen for all parameters (often the single combination of settings that leads to the best performance for the tuning set is used). Next, the system is retrained using both the training set and tuning set, and then the system is tested on the test set to get an estimation of its future performance. (The results for the tuning set itself is not a fair estimate, since this is the exact data for which the system has been optimized.) When using this methodology, it is important that all three subsets of the labeled data are distinct and non-overlapping; otherwise, there would be times when the system is being tested on data that is also used for training, and this would give the system an unfair advantage and make the results inaccurate. If the work is being done for pragmatic benefit as opposed to research (i.e. there will actually be future documents to which the final system is applied), then after the final evaluation based on the test set, the system can be retrained using the entire labeled corpus, including the test set, since it is generally best to use as much training data as possible.

Another common methodology for choosing settings of parameters involves the use of *k-fold cross validation* (Mitchell, 1997); to implement this strategy, the training set needs to be divided into k non-overlapping subsets, or folds, usually of approximately equal size. For each possible combination of parameters, the system is repeatedly trained on $k-1$ subsets of the training data and tested on the remaining subset of the training data; this is repeated k times, with each of the k subsets used for testing once. For each combination of parameters, the results of the k cross validation experiments are averaged (perhaps using a weighted average if the subsets are not of equal size). These results are used to choose a single combination of settings (often the one that leads to the best average performance). Then the system can be retrained with the chosen settings using the entire training set, and then tested on the test set to obtain an estimation of future accuracy. It is, of course, important that the test set does not overlap with the training set for the same reason mentioned in the discussion of the other methodology for choosing parameters (involving the use of a tuning set). Also, as was the case with that methodology, after the test set is used to estimate future performance, the system can be retrained using the entire labeled corpus.

When performing cross validation experiments, the minimum possible value of k is 2 and the maximum possible value of k is equal to the number of documents in the training set. When k is exactly equal to the number of documents in the training set, every round of training uses all but one of the documents, and the final document is used for testing; this is also known as leave-one-out cross validation. Larger values of k mean more experiments and therefore more time, but it is unclear if increasing k leads to an expected increase in accuracy of future performance. See (Kohavi, 1995) for a study of the effects of varying k for several tasks. (This work also compares cross validation to the tuning set methodology, which they call the holdout method, and also to a method known as the bootstrap (Efron and Tibshirani, 1993), which I do not discuss in this chapter.)

In my own work, I have preferred the cross validation methodology to the tuning set methodology, although I have never actually compared the two. In Appendix G, I describe my use of cross validation to select settings for several optional parameters of a system I have developed. I have used three-fold cross validation, choosing the low value of three for k mainly because I was, at the time, dealing with many possible combinations of parameter settings and there was an efficiency concern.

After settings for all optional parameters of a system are determined, either using cross validation experiments or experiments with a tuning set, the system is trained, and then it is ready to be applied to future documents. Most text categorization approaches, including all of those described in Section 2.6, can be applied to new documents one at a time. This may be required of certain systems dealing with new documents that arrive one at a time and must immediately be categorized. An obvious example of this type of task would be the classification of e-mail as *spam* versus *not spam*. However, there are certain methods (or particular variations of methods) that require access to the entire test set at one time (or at least a large number of test instances at one time). For example, the Pcut method for allowing the Rocchio approach to be used with binary categories (both Pcut and Rocchio are described in Section 2.6.1.1) needs to measure similarity between all test documents and all categories before predictions can be made.

2.6 Machine Learning Approaches for Text Categorization

Over the years, many machine learning approaches have been developed for automatic text categorization. This section describes some of the more common methods. The three approaches described in Section 2.6.1 (Rocchio/TF*IDF, K-Nearest Neighbor, and Naive Bayes) involve basic techniques that have been around for a relatively long while and are often used as baselines against which newer techniques are compared. These approaches are described to the point that a programmer should be able to implement them after reading the section. The three approaches described in Section 2.6.2 (Probabilistic Indexing, Maximum Entropy, and Support Vector Machines) involve more recent and mathematically complex techniques. Most of the text categorization literature points to these as being "better" than the basic approaches, although I have found that with some modification, approaches stemming from the basic ones are very competitive. For example, the system I describe in Chapter 4 and in (Sable, McKeown, and Hatzivassiloglou, 2002) uses density estimation to improve the results of a Rocchio/TF*IDF approach, and the system I describe in Chapter 5 (an earlier version is described in (Sable and Church, 2001)) uses bins to smooth a Naive Bayes approach; for the data sets of the corpus I have created (described in Section 3.1), these systems are very competitive with all of the alternative I have tested.

2.6.1 A Few Basic Approaches

The three approaches described in this section are based upon three simple ideas. The Rocchio/TF*IDF approach uses a bag of words representation (as described in Section 2.4) involving TF*IDF word weights (as described in Section 2.4.2) not only for documents, but also for categories. Given a document that needs a prediction, it is compared to all categories, and assigned to the category that is the most "similar", based on a simple metric such as the cosine measure or dot product. The K-Nearest Neighbor approach compares each new document to labeled documents in the training set. The k training documents that are the most similar are observed, and the majority category based on a weighted average of these documents is assigned to the new document. Naive Bayes does not use weights for words as described in Section 2.4.2; instead, category-specific weights are empirically estimated for all words based on the training set. The training documents are used to estimate the probability of seeing each possible word in each possible category. Given a new document, these probabilities are combined to estimate the probability that such a document would occur in each possible category, and the category with the highest probability is assigned to the document. (These short descriptions assume mutually exclusive categories; in the subsections ahead, I also include brief discussions of how the techniques can be modified to deal with binary categories.)

2.6.1.1 Rocchio/TF*IDF

The Rocchio/TF*IDF approach stems from Rocchio's method of *relevance feedback* (Rocchio, 1971), a technique used to improve information retrieval performance (a task defined in Section 2.3). This technique allows a user to rate documents initially retrieved for some given query as *useful* or *not useful*, and based on these evaluations, a second pass is implemented. The approach has since been generalized to apply to text categorization tasks with multiple categories. For example, see (Sable and Hatzivassiloglou, 1999) and (Sable and Hatzivassiloglou, 2000) which describe an augmented, Rocchio/TF*IDF system that I and a fellow researcher have developed and applied to the categorization of images.

As discussed in Section 2.4, most text categorization systems represent documents using a bag of words approach in which each document is represented as a vector of weighted words. Then any document d can be represented as $d = [\lambda_{t_1}, \lambda_{t_2}, ..., \lambda_{t_{|T|}}]$, where λ_t is the weight of a single term t from the set of all terms T. Bag of words representations use single words as terms, although exactly what constitutes a word can vary as described in Section 2.4.1. The most common weighting scheme for words is to multiply each word's term frequency (TF) by its inverse document frequency (IDF) as explained in Section 2.4.2. The Rocchio/TF*IDF approach towards text categorization represents not only documents, but also categories with such vectors. The vector representation for a category is a combination (generally the sum) of the vectors representing the training documents that are assigned to the category; often these category vectors are normalized. Each such vector can be thought of as a *centroid* of the category.

Once vectors are set up for all documents and categories, a new document can be compared to all possible categories and assigned to the category yielding the highest similarity. More formally, let d be a new document and C be the set of all possible categories (represented with vectors as previously described). Then the category predicted for d is simply:

$$\underset{c \in C}{argmax}[cosine(d, c)] \tag{2.3}$$

Alternatively, the dot product can be used to compare d with each possible c instead of the cosine measure. Applying the cosine metric to two vectors is equivalent to applying the dot product and then dividing by the product of the lengths of the two vectors. In other words:

$$cosine(d, c) = \frac{d \cdot c}{|d||c|} \tag{2.4}$$

The length of the document vector d does not affect results, since for each document, we are choosing the category with the highest similarity, and the length of d is constant. Using the cosine metric instead of the dot product is therefore equivalent to normalizing the category vectors, so as not to give too much weight to larger categories (i.e. categories with more training examples). If the category vectors have already been normalized, the two metrics give equivalent results. (See Appendix G for a discussion of the effects of normalization and several other optional components on the performance of a Rocchio/TF*IDF system applied to multiple sets of categories.)

Things are a little more complicated when dealing with binary categories, since a document may be assigned to no categories or multiple categories. In this case, there are several standard methods of converting similarity scores to YES/NO decisions for every possible document/category pair. One, known as Scut (Yang, 1999), involves determining the optimal threshold for each category based on training data. Another method, known as Pcut (Yang, 1999), involves measuring the percentage of training documents that fall into each category in the training set and assuming a similar percentage falls into the category in the test set. Therefore, for each category, we assign the category to the x test documents with the highest similarity, where x is chosen based on the training set. One drawback of Pcut is that it can not be applied to text categorization tasks involving new documents that arrive one at a time if each must be immediately labeled independently of others. For both Scut and Pcut, normalization of document vectors (as opposed to category vectors) becomes an issue. A third method is to create, for each category, a separate category consisting of all documents not in the actual category. To obtain a YES/NO decision for the actual category, we compare the similarity score of a test document and the actual category to the similarity score of the test document and the created category, thus converting our multi-label, binary categorization task to a set of categorization tasks each with two mutually exclusive categories. All of these methods have drawbacks, and it seems to me that Rocchio/TF*IDF generally does not perform as well for binary text categorization tasks as it does for tasks involving mutually exclusive categories. See, for example, (Yang, 1999), in which Rocchio/TF*IDF underperforms most other methods tested for a binary categorization task involving the commonly used Reuters corpus (described in Section 2.8). In Chapter 4, on the other hand, I show that my own Rocchio/TF*IDF system performs quite well for the data sets in my own corpus, involving mutually exclusive categories, especially when aided by the density estimation technique described in that chapter.

2.6.1.2 K-Nearest Neighbor

The K-Nearest Neighbor (kNN) approach towards text categorization is an example of an instance-based learning method (also known as an example-based method, a memory-based method, and even a lazy learning method) (Mitchell, 1997; Sebastiani, 2002). The training for such methods basically just consists of recording which training examples fall in each possible category. (Of course, reading in the training documents also requires a system to convert them to the appropriate bag of words representation, assuming the system does not require this representation as input in the first place.) When future documents arrive (or a test set is evaluated), the new documents are compared to those in the training set.

In order to predict the category of a document, a system must first determine which training instances are the "closest" to this document. The closeness of one document to another is typically determined by Euclidean distance. In other words, if $d1$ and $d2$ are two documents such that $d1 = [\lambda_{t_{(1,1)}}, \lambda_{t_{(1,2)}}, ..., \lambda_{t_{(1,|T|)}}]$ and $d2 = [\lambda_{t_{(2,1)}}, \lambda_{t_{(2,2)}}, ..., \lambda_{t_{(2,|T|)}}]$ (see Section 2.4), and $D_{(d_1,d_2)}$ represents the distance between $d1$ and $d2$, then we have:

$$D_{(d_1,d_2)} = \sqrt{\sum_{i=1}^{|T|}[\lambda_{t_{(1,i)}} - \lambda_{t_{(2,i)}}]^2} \tag{2.5}$$

Once distances are computed between the new document and all training documents, the k closest documents (and thus the k documents with the smallest distances according to the above formula) are retrieved.[7] The specific value of k that a system uses can be set arbitrarily, but it is generally better to determine an appropriate value of k based on cross validation experiments or experiments with a tuning set (see Section 2.5).

Once the k closest training documents are retrieved, they are used to predict the category for the new document. The simplest way to do this is to assign the new document to the category that is the most common among the k retrieved training documents. (If the task involves binary categories, an alternative is to assign the new document to any category that is assigned to at least half of the k documents, or separate cutoffs can be determined for every category based on cross validation

[7]An alternative method of finding the closest training documents is to use the cosine metric or dot product to measure the similarity between the new document and all training documents, in which case the k documents with the highest similarities are retrieved.

experiments or experiments with a tuning set.) More common than taking this very simple approach is to weight each of the k training examples inversely proportional to its distance from the new document.[8]

For a given document d, the kNN approach is using the k retrieved neighbors from the training set to calculate a score for every possible category c; this score is equal to the proportion of the neighbors that are assigned the label c. As mentioned in the last paragraph, in calculating this proportion, it is common to weight the neighbors inversely proportional to their distance from d. A separate score can be computed for each category based on the (weighted) percentage of close training documents that have been assigned to the category, and this score can be viewed as an estimate of the probability that the document belongs to the category. More formally, let d_i be the i^{th} training document out of the k training documents selected as described above, and let $I_{(d_i,c)}$ be 1 if d_i belongs to category c and 0 otherwise. Given a document d, the score that is calculated for some specific category c is thus:

$$S_{(d,c)} = \frac{\sum_{i=1}^{k} I_{(d_i,c)} \frac{1}{D_{(d,d_i)}+\epsilon}}{\sum_{i=1}^{k} \frac{1}{D_{(d,d_i)}+\epsilon}} \tag{2.6}$$

Note that the numerator and denominator in the above formula are the same except for the $I_{(d_i,c)}$ term, so that documents that belong to category c contribute to both the numerator and denominator, and documents that do not belong to category c contribute only to the denominator. The epsilon in the formula is just an arbitrary, very small constant to avoid infinities in the case that there is some training document with a representation that exactly matches that of d (in which case the distance is 0).

The document d can then be assigned to the category with the maximum

[8]If training documents have been selected based on their similarities to the new document as opposed to their distances from the new document, then each of the k training documents should instead be weighted directly proportional to its similarity to the new document.

score, that is:

$$\underset{c \in C}{argmax}[S_{(d,c)}] \tag{2.7}$$

If we are dealing with binary categories, the document can instead be assigned to all categories with a score greater than or equal to 0.5, or separate cutoffs can be determined for each category using cross validation experiments or experiments with a tuning set. (The reason 0.5 is a natural cutoff is because the score $S_{(d,c)}$ can be thought of as an estimated probability that d belongs to category c.) When dealing with mutually exclusive categories, the scores, or probability estimates, always add up to exactly 1; since all documents must be assigned to exactly one category, it should be assigned to the category with the highest score, even if no score is above 0.5. When dealing with binary categories, each individual category is assigned a probability ranging from 0 to 1, but there is no further restriction on the sum of these probabilities, since the categories are independent, and a document may belong to more than one category or none at all.

In some recent and well-cited studies by Yang (Yang, 1999; Yang and Liu, 1999), kNN has performed very well on text categorization tasks involving the often used Reuters data set (to be described in Section 2.8). However, I have found in my own work that kNN usually underperforms other methods, even the other basic ones. The specific properties of the data and the categories involved with text categorization tasks vary highly, and this determines which methods perform the best. (See Section 2.9 for a detailed discussion concerning the ways that such properties can vary and my own insights into how this might affect various systems.)

2.6.1.3 Naive Bayes

The Naive Bayes approach towards text categorization (Lewis, 1998) is a probabilistic approach that attempts to estimate the probability of each possible category given a document. More formally, let d be a new document and C be the set of all possible categories. Then, for all $c \in C$, a Naive Bayes system calculates an estimate of $P(c|d)$ and choose the category for which this estimate is the largest. An application of Bayes' theorem tells us that:

$$P(c|d) = \frac{P(c)}{P(d)} \times P(d|c) \tag{2.8}$$

The denominator in the right side of this equation is the same for all categories and can therefore be dropped if the sole goal is to obtain a correct classification. If probability estimates are actually desired for each category, the final probability estimates obtained without using this term can easily be normalized, since the probabilities for all possible categories must add up to one, assuming that we are dealing with mutually exclusive categories. After dropping the $P(d)$ term from the above equation, a Naive Bayes system assigns a document d to:

$$\underset{c \in C}{argmax}[P(c) \times P(d|c)] \tag{2.9}$$

The term $P(c)$ represents the *a-priori* probabilities of category c, and this is often estimated as the percentage of the training set that belongs to the category. If, for some reason, it is suspected that the training set is not representative in this manner, it may be better to assign equal initial likelihoods to each category (in which case this term may be dropped from the equation all together without affecting results).

In order to estimate $P(d|c)$, a Naive Bayes classifier makes an *independence assumption*, which views every coordinate in the bag of words representation of d as an independent random variable. In other words, it is assumed that the likelihoods of various possible weights for any given word in d is not affected by the weights of other words. Expressed in yet another way, the assumption claims that the occurrence of one word in a text does not affect the chances of seeing any other word. Of course, in reality, this assumption is constantly violated. Seeing one word in a text makes it more likely that certain related words will appear, and it may be less likely that other specific words will appear. It is for this reason that the word "naive" is part of description of such a classifier. Although this assumption surely throws off the estimated probabilities (this is discussed further later in the section), the prediction of the system will still be correct as long as the correct category obtains a higher estimate than all other categories. Using the notation from Section 2.4, let T be the set of all possible terms (words) and λ_t be the weight of a single term $t \in T$. Then any document d can be represented as $d = [\lambda_{t_1}, \lambda_{t_2}, ..., \lambda_{t_{|T|}}]$. Given the independence assumption just discussed, we can say that:

$$P(d|c) = \prod_{t \in T} P(\lambda_t|c) \tag{2.10}$$

Here, $P(\lambda_t|c)$ represents the probability that term t has weight λ_t in a document of category c.

Naive Bayes systems do not use a TF*IDF weighting system to represent a document, but instead words are commonly weighted with binary values (1 if a word is present and 0 otherwise). This variation of Naive Bayes is known as the *binary independence* model (Sebastiani, 2002), and it is heavily promoted in (Robertson and Jones, 1976) in which the authors use this model for relevance feedback (a term also mentioned at the start of Section 2.6.1.1) in an information retrieval context. For the most part, weights of 0 are not very informative; in other words, the absence of a word in a document is not generally indicative of any particular category. Weights of zero are therefore often ignored by Naive Bayes systems, meaning that only words that actually appear in a document are considered. Letting W be the set of all distinct words in a given document d, a Naive Bayes then assigns the document to:

$$\underset{c \in C}{argmax}[P(c) \times \prod_{w \in W} P(w|c)] \tag{2.11}$$

Here, $P(w|c)$ represents that probability that the word w occurs at least once in a document of category c; this can be easily estimated based on the training data as the percentage of those training document belonging to category c that contain at least one instance of the word w. (Some Naive Bayes implementations rely on a more complex estimates of $P(w|c)$; e.g. see (Joachims, 1997) which describes a Naive Bayes implementation that uses the Laplace estimator, which has the advantage that it never estimates probabilities to be zero.) For computational reasons, it is often more efficient to work with log likelihoods (i.e. add logs of probabilities) than it is to multiply probabilities directly. Since the log of a product is equal to the sum of the log of its parts, it is therefore equivalent for a Naive Bayes system to assign the document to:

$$\underset{c \in C}{argmax}[\log_2 P(c) + \sum_{w \in W} \log_2 P(w|c)] \tag{2.12}$$

To obtain the actual probability estimate for the category, if it is desired, the system can raise the logarithmic base being used (shown here as 2) to the power of the final sum for the category.

Many Naive Bayes implementations loop through all instances of all words in the documents as opposed to only considering distinct words, since the repetition of a word does tend to indicate importance (see (Joachims, 1997), once again, as an example). Equivalently, some Naive Bayes implementations only loop through distinct words but raise the probability of seeing each word in a category to a power equal to the term frequency of the word in the current document (e.g. see (McCallum and Nigam, 1998)). These implementations are therefore actually making a larger assumption than the independence assumption discussed above; this assumption is that the observation of a word in a document not only has no effect on the chances of seeing other words, but also that it has no affect the chance of seeing the same word with repeat occurrences. This has clearly been shown to be false for content words by Church (2000), in which the author shows that for content words, once a word appears in a document, the chance of the word occurring again is generally closer to 1/2 than p, where p is the *a-priori* probability of seeing the word occur in a document (generally much smaller than 1/2 for content words).

Another method that a Naive Bayes system can use to weight repeated words higher without making this stronger assumption is to consider the term frequency of words, and to estimate probabilities, based on the training set, of seeing each possible word with various occurrence counts in each possible category. In other words, for a given word w and category c, the training documents belonging to c are used to estimate probabilities of seeing w one time, two times, three times, etc. There can be some maximum cutoff above which counts are not distinguished; for example, the system may estimate a single probability of seeing w four or more times in a document of category c. An example of a system using this approach is BINS, my own bin-based system discussed in Chapter 5 (an older version is discussed in (Sable and Church, 2001)), which can be thought of as a generalized version of a Naive Bayes system. (BINS estimates probabilities for groups of related words instead of individual words in order to smooth estimates that may be inaccurate for words with scarce evidence.)

While Naive Bayes often produces good categorization results, the probability estimates themselves are usually not accurate. In my own research, I have noticed that Naive Bayes techniques usually suggest an estimated probability of over 99%, and often over 99.9%, for the favored category, while the estimated probabilities for all other categories are therefore very close to zero (since my work has typically involved mutually exclusive categories). In order for these probabilities to be accurate, over 99% of the predicted categories would have to be correct, but for the tasks I have been dealing with, accuracy rates of around 80% for Naive Bayes systems are more common. These inaccurate probability estimates are likely due to the falsehood of the independence assumptions that have been made; Naive Bayes systems are inherently multiplying together probabilities (or adding log likelihoods) that are not truly independent, and therefore exaggerating their magnitudes. In

(Bennett, 2000), evidence is presented indicating that Naive Bayes systems typically produce probabilities close to 0 or 1 for text categorization tasks involving binary categories, and the author begins to discuss methods that could be used to recalibrate the probabilities. In a later paper (Bennett, 2002), he introduces methods of fitting asymmetric Gaussian and Laplace distributions to the output of a Naive Bayes classifier in order to improve the probability estimates. My method of using density estimation (explained in Chapter 4) could also potentially be used for this purpose.

Naive Bayes is a very popular method in the text categorization and information retrieval literature. In recent, extensive studies by Yang (Yang, 1999; Yang and Liu, 1999), a Naive Bayes system has not performed as well as some others tested for text categorization tasks involving the often used Reuters data set (to be described in Section 2.8). However, I have found that Naive Bayes performs well for data sets in a corpus I have created (described in Section 3.1), and in Chapter 5, I show that my BINS system, which can be thought of as a generalized version of a Naive Bayes system, outperforms all other systems tested for two of these data sets. Section 2.9 discusses some reasons why certain approaches may perform very well for some text categorization tasks and not for others.

2.6.2 A Few Advanced Approaches

Each of the approaches discussed in the previous subsection is based on a simple principle. The Rocchio/TF*IDF approach compares a new document to each possible category and chooses the category with the most similar centroid. The kNN approach compares a new document to all training documents and chooses the category that is assigned to the most similar training documents. The Naive Bayes approach estimates probabilities of words occurring in categories, and given a new document, it chooses the category that is most probable to contain the words in the document. Each of these approaches is intuitive and popular in the text categorization and information retrieval literature. While they are considered basic, and are often just used as baselines against which advanced methods are compared, there are, at least, certain tasks for which they perform very competitively.

The approaches discussed in this subsection are more complex. In the literature, authors who describe them tend to consider them *better*. Papers describing them usually show improvement using these techniques in comparison to the basic ones. To some extent, this may be a little misleading. There are very few public text categorization corpora (discussed in Section 2.8), with the Reuters corpus (also discussed in Section 2.8) dominating the literature. I have found in my own research that no system does the best on all text categorization tasks, and almost all well-known approaches, including the basic ones discussed in Section 2.6.1, perform quite well for at least some tasks (I try to explain some of the likely reasons why this occurs in Section 2.9). In addition, to effectively compare systems, it is very important to use the same evaluation metrics (discussed in Section 2.7). Since different evaluation metrics may favor different systems, it is important that researchers choose the metrics *in advance*, as opposed to choosing a metric that favors the approach they are hoping does well after obtaining their results. Finally, it should be noted that while published papers in the field seem to show that advanced methods outperform the basic ones, I feel that there is something unrepresentative here. Researchers who find that the advanced methods they are exploring do not outperform basic ones are certainly less likely to have their findings published, since this is not considered interesting.

That being said, it is important to understand advanced techniques, as they comprise a major part of the text categorization literature, and, at the very least, there is no reason to doubt that they tend to do at least as well as the basic approaches on average. I have chosen three advanced approaches to describe in this section, partly because of their abundance in the text categorization literature, and partly because they are approaches used by systems that comprise the publicly available Rainbow package (described in Section 2.8), which I use throughout this thesis to compare against my own systems. These approaches are Probabilistic Indexing, Maximum Entropy, and Support Vector Machines. I do not describe these approaches to the same level of detail as the basic approaches described in Section 2.6.1; that would be beyond the scope of this chapter. Instead, I explain the main ideas in order to give a basic understanding of these algorithms.

2.6.2.1 Probabilistic Indexing

The Probabilistic Indexing approach towards text categorization stems from Fuhr's Probabilistic Indexing paradigm (Fuhr, 1988), which Fuhr originally intended for relevance feedback in an information retrieval context. Joachims (1997) generalizes the technique for text categorization (he assumes categories are mutually exclusive, as I do throughout most of this chapter), and he calls his classifier PrTFIDF (because he considers it to be a probabilistic version of a Rocchio/TF*IDF classifier). Like Naive Bayes, Probabilistic Indexing is a probabilistic approach (not surprisingly given the name) that attempts to estimate the probability of each possible category given a document. The description of this approach will likely seem much more similar to Naive Bayes than to the Rocchio/TF*IDF approach described in Section 2.6.1.1; however, Joachims (1997) shows that Probabilistic Indexing and Rocchio/TF*IDF are actually similar in that the two techniques yield identical categorization results under certain conditions. (These conditions involve assumptions that don't exactly hold in practice, and for the experiments discussed in that paper, a Probabilistic Indexing implementation outperforms a Rocchio/TF*IDF implementation and performs about the same as a Naive Bayes implementation.)

The Probabilistic Indexing paradigm distinguishes between a document, d, and its representation. Let d be a document that needs to be categorized, and Θ be a function that maps a document to its representation. Then, for all $c \in C$ (where C is the set of possible categories), a Probabilistic Indexing system calculates an estimate of $P(c|d, \Theta)$ and choose the category for which this estimate is the largest.

The tricky step comes with the choice of Θ, which is not a deterministic function. Every document is represented by a single word that is chosen randomly from the bag of words representation of the document. In other words, letting W be the set of all words in a document d (that is, all instances of all words, not just distinct words), Θ randomly maps d to a single word chosen from W. So we can now rewrite the probability we are trying to maximize as:

$$P(c|d, \Theta) = \sum_{w \in W} [P(c|w) \times P(w|d, \Theta)] \tag{2.13}$$

We can rewrite $P(c|w)$ with an application of Bayes' theorem:

$$P(c|w) = \frac{P(c) \times P(w|c)}{P(w)} = \frac{P(c) \times P(w|c)}{\sum_{c' \in C}[P(w|c') \times P(c')]} \tag{2.14}$$

We are now at a stage that everything can be nicely estimated based on the training data. $P(w|d, \Theta)$ is simply $TF(w, d)$ / $|W|$ (where $TF(w, d)$ is the term frequency of w in document d as described in Section 2.4.2, and $|W|$ is the number of words in the document). $P(c)$ for any category c can be estimated as the percentage of training documents that belong to the category. Finally, $P(w|c)$ for any category c can be estimated by dividing the number of times w occurs in training documents belonging to the category (i.e. the sum of the word's term frequencies over all documents in the training set belonging to the category) by the total number of occurrences of all words in the same training documents (i.e. the sum of all words' term frequencies over all documents in the training set belonging to the category). These estimates allow the system to compute a probability estimate for each possible category, and the category with the highest probability is selected.

Probabilistic Indexing systems are less common in the text categorization literature than the other two advanced approaches I discuss in this section. However, of the six systems that comprise the publicly available Rainbow package (described in Section 2.8), I think that the Probabilistic Indexing system has most consistently yielded impressive results. In fact, for three of the four data sets that are part of the corpus I have created (described in Section 3.1), the Probabilistic Indexing system performs the best of these six systems. The Probabilistic Indexing system also performs favorably against my own systems; it outperforms my density estimation system described in Chapter 4 for all four data sets (although my density estimation system provides more useful probability estimates), and it outperforms my BINS system described in Chapter 5 for two of the four data sets (my BINS system with the appropriate settings outperforms all others tested for the other two data sets).

2.6.2.2 Maximum Entropy

Maximum Entropy is a technique that has proven quite successful for a variety of NLP applications (Ratnaparkhi, 1998; Berger, Pietra, and Pietra, 1996), and it has also been applied to text categorization tasks (Nigam, Lafferty, and McCallum, 1999). Like Naive Bayes and Probabilistic Indexing, Maximum Entropy is a probabilistic approach, and in the domain of text categorization, probabilities are estimated for each possible category that might apply to a given document. The general principle behind Maximum Entropy approaches is to assume nothing, and to consider all possibilities equally likely unless there is evidence otherwise. In other words, the goal is to maximize entropy, which can be thought of as a measure of uncertainty, subject to constraints derived from empirical evidence. A Maximum Entropy system starts off assuming that all possible outputs are equally likely, and then it repeatedly updates all probabilities as new evidence is observed, adhering to certain constraints representing characteristics of the training data. When applied to text categorization tasks, the possible outputs are the categories, and the constraints are generally in the form of expected word counts for documents of each possible category.

Let's say we are dealing with a text categorization task that I discuss later in this thesis, that of categorizing news documents into the categories *Struggle*, *Politics*, *Disaster*, *Crime*, or *Other* (defined to be mutually exclusive). Since there are five categories, a Maximum Entropy system assumes that each has a probability of one fifth. In other words, $P(\mathit{Struggle}) = P(\mathit{Politics}) = P(\mathit{Disaster}) = P(\mathit{Crime}) = P(\mathit{Other}) = 0.20$.

Now let's say we know one thing about a document d_1, that it contains the word "earthquake". Furthermore, we have one constraint based on the training data expressing that there is an 80% chance that a document with the word "earthquake" in it belongs to the *Disaster* category. Knowing nothing else, the way to maximize entropy is to assign $P(\mathit{Disaster}) = 0.80$ and $P(\mathit{Struggle}) = P(\mathit{Politics}) = P(\mathit{Crime}) = P(\mathit{Other}) = 0.05$. Let's also say for some other document, d_2, we know only that it contains the word "president", and we also have a constraint expressing that there is a 70% chance that a document with the word "president" belongs to either the *Struggle* category or the *Politics* category. Knowing nothing more and avoiding all assumptions, the way to maximize entropy is to assign $P(\mathit{Struggle}) = P(\mathit{Politics}) = 0.35$ and $P(\mathit{Disaster}) = P(\mathit{Crime}) = P(\mathit{Other}) = 0.10$. These cases are both intuitive, and the probability assignments are obvious. However, let us say that for some other document, d_3, we are aware of two facts - that it contains the word "earthquake" and that it contains the word "president". What do we do? This time, the answer is not obvious.

Maximum Entropy provides a way of combining constraints that overlap. First, a way of expressing features of documents is needed. Following the notation in (Nigam, Lafferty, and McCallum, 1999), the features they use to apply Maximum

Entropy to a text categorization task are:

$$f_{w,c'}(d,c) = \begin{cases} 0 & \text{if } c \neq c' \\ \frac{TF(w,d)}{|W|} & \text{otherwise} \end{cases} \tag{2.15}$$

In this equation, (d, c) represents a document/category pair while (w, c') represents a word/category pair. $TF(w, d)$ is the term frequency of the word w in document d, $|W|$ is the set of all words in d (that is, all instances of all words, not just distinct words), and $|W|$ is the number of words in d. So, in expressing the features for some given document, words that are not in the document do not add weight to any category (since $TF(w, d)$ will be zero), while words that are in the document add weight to the category to which the document belongs.

In order to estimate the probability of a category given a document, the features of the document are combined according to the following equation:

$$P(c|d) = \frac{1}{Z(d)} exp(\sum_i \lambda_i f_i(d,c)) \tag{2.16}$$

Here $Z(d)$ is simply a normalizing factor that ensures the probabilities for the various possible categories sum to 1:

$$Z(d) = \sum_c exp(\sum_i \lambda_i f_i(d,c)) \tag{2.17}$$

When applied to a text categorization task using the features defined earlier, the variable i is actually looping through all of the word/category pairs, since each word/category pair represents a distinct feature of a document. Only words that exist in a document need to be considered, because the value of features corresponding to words that do not exist in the document are zero, and they do not contribute to the sum used in the exponent. This leaves λ_i, which needs to be estimated for each possible feature based on the training data. These estimates take into consideration that expected values for features in a model distribution should match what has been observed in training data.

The trick, then, is to estimate all values of λ_i, adhering to the constraints (the counts seen in the training data), in such a way as to maximize the entropy of the system. The algorithm that Maximum Entropy systems use to do this is called *improved iterative scaling* (IIS). I am not going to give a formal description of this procedure. Instead, I try to provide a very basic understanding of the principles involved. For a formal description of improved iterative scaling and how it is used to estimate the parameters for a Maximum Entropy system, I suggest reading (Berger, Pietra, and Pietra, 1996) or (Ratnaparkhi, 1997), or for a description that specifically addresses the use of improved iterative scaling and Maximum Entropy for a text categorization task, I suggest (Nigam, Lafferty, and McCallum, 1999).

Improved iterative scaling starts with any initial distribution for parameters (e.g. every λ_i can be assigned to 0). It then initiates a hill-climbing approach that predicts categories for the training documents and, based on the results (some of which are correct and some of which are not correct), finds an incrementally more likely set of parameters. It has been shown that the space through which improved iterative scaling is searching (each point in this space represents a possible set of values for the parameters) contains no local maxima; thus, every iteration moves the parameters closer to the global maximum entropy solution. Typically, Maximum Entropy systems iterate until the values seem to converge, or they iterate some fixed number of times.

In (Nigam, Lafferty, and McCallum, 1999), the Maximum Entropy system they test beats a Naive Bayes system for two out of three text categorization tasks. In my own research, I have seen that the Maximum Entropy system I have tested (that which is part of the Rainbow package discussed in Section 2.8) generally falls somewhere in the middle of the pack of tested systems. Still, Maximum Entropy has had a lot of success in several other areas of NLP, and it is certainly a technique that is worth considering.

2.6.2.3 Support Vector Machines

The Support Vector Machine (SVM) approach towards text categorization is based on the principle of Structural Risk Minimization (Vapnik, 1995). To fully explain the details of SVMs is far beyond the scope of this chapter. For a complete and formal description of SVMs, I would recommend the book by Cristianini and Shawe-Taylor (2000) or the tutorial by Burges (1998). Here, I provide an overview of the main ideas that should result in an intuitive understanding of how SVMs work. SVMs have been successful in many areas of machine learning, with applications including handwritten digit recognition, speaker identification, face detection in images, and bioinformatics (the two recommended sources contain references to papers discussing the use of SVMs for all of these purposes). Of course, the reason I am including a discussion of SVMs in this chapter is that they have also been successfully applied in the field of text categorization (Joachims, 1998; Elworthy, 1998; Drucker, Wu, and Vapnik, 1999; Yang and Liu, 1999; Dumais and Chen, 2000).

SVMs require that documents be represented as vectors. In the case of text categorization, bag of words approaches as described in Section 2.4 are typically used for the representation, so each dimension of the vector represents a single word and the value of that dimension represents the weight of the word. The description of SVMs in this section, however, is more general, and it is better to think of each document as being represented by any vector of real values. Unlike the other approaches I have discussed, SVMs are, in principle, really geared towards binary categorization tasks, so at first, I assume that this is what we are dealing with; later I discuss the possibility of using SVMs when dealing with mutually exclusive categories. Let's say each document vector consists of n dimensions. Then, in its simplest form, an SVM system tries to find an $n-1$ dimensional *hyperplane* that separates the training documents that belong to the category being considered from those that do not. If such a hyperplane exists, the training documents are said to be linearly separable. When a new document is encountered (also represented in vector notation), the system checks on which side of the hyperplane the document falls, and uses this to predict whether or not the document belongs to the category. Usually, if there are any such hyperplanes that separate the positive examples from the negative examples, then there are infinitely many, and the SVM system chooses the one that maximizes that margin, which represents the shortest distance between the hyperplane and any training example.

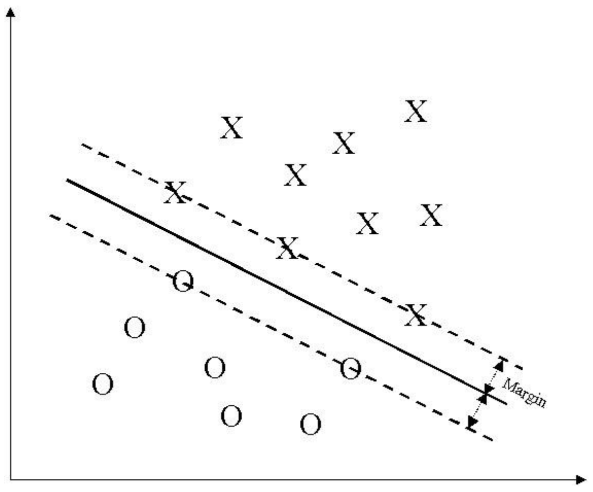


Figure 2.1: The bold line represents the separating hyperplane with the maximum margin.

Figure 2.1 depicts a simple case in which each document is represented by a vector of two real values. It is easy to think of this in geometric terms, with each document being represented by a point in a two-dimensional Euclidean plane. If the document belongs to some particular binary category (there is no need to actually specify the category for this discussion), it is represented by an ‘X’, and if not, it is represented by an ‘O’. The two dashed lines and the solid line in the middle are all separating hyperplanes, and any other parallel line in between the dashed lines would also be a separating hyperplane. But the solid line maximizes the margin. Moving or rotating that line at all would lessen the distance between the line and at least one of the training vectors that fall on the dotted lines.

The vectors that fall on the dotted lines (and therefore are a distance of exactly one margin from the optimal separating hyperplane) are called *support vectors*; in a sense, they are the only training documents that matter. Removing one of these documents might (but would not necessarily) change the solution. Removing any other document would not change the solution. Notice, then, that many training documents do not affect the system at all. If this seems unintuitive, think of the documents representing the support vectors as the most difficult to predict, since they are the closest to the borderline. The trained system has learned a hyperplane that is close to these difficult to predict documents; the documents that are much further away are obvious.

Of course, in actual practice, it is often the case that there are no separating hyperplanes at all. Sometimes, this is just because of a few training vectors that make such a solution impossible. In other words, there may still be a hyperplane that separates most of the positive examples from most of the negative examples. Such a hyperplane, used in the same manner as a separating hyperplane, would still likely be an accurate predictor for future documents. SVM implementations can take this into account, often finding an optimal hyperplane that is not actually a separating hyperplane.

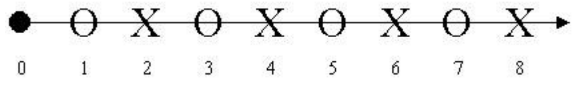


Figure 2.2: In this case, a separating hyperplane would have to be a single point, but no single point does a good job separating positive examples from negative examples.

Sometimes, though, even this is not good enough, and there are cases in which no hyperplane in the original feature space does a good job distinguishing positive examples from negative examples. Figure 2.2 graphically depicts an even simpler case than the last example, in which each document is represented by a vector of only one real value. (That is, this case is simpler in terms of geometry, but it does not have as simple a solution, as we shall see.) Each document is represented by a point on a number line. Again, any document that belongs to the category being considered is represented by an 'X' and any document that does not belong to the category is represented by an 'O'. Here, though, a difficulty arises. A separating hyperplane would be a single point separating the positive examples from the negative examples. But in this case, points with an even value tend to belong to the category being considered, and points with an odd value tend not to belong to the same category. No separating hyperplane does a good job separating the positive examples from the negative examples.

In cases such as this, SVM systems map the vector representations of the documents to a different feature space, usually one of higher dimension. The hope is that in this destination feature space, the positive and negative examples of the category are linearly separable (or, at least, closer to it). Then, an optimal hyperplane can be found in this new feature space. When a new document is encountered, the vector representation of the document is also mapped to the new feature space; then the system observes on which side of the optimal hyperplane in this new space the mapped vector falls, and makes a prediction about the category being considered accordingly.

Looking back at Figure 2.2, let's say we have a function that maps each vector from the one-dimensional feature space to a two-dimensional feature space such that the x-coordinate in the new feature space has the same value as the single coordinate in the original feature space and the y-coordinate in the new feature space is +1 or -1 depending on whether the single coordinate in the original feature space is even or odd. Figure 2.3 graphically depicts the same document

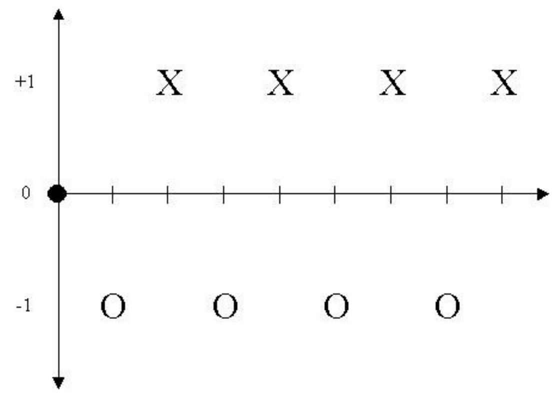


Figure 2.3: After mapping the vectors from the previous figure to a new feature space, the positive examples are linearly separable from the negative examples.

vectors depicted in Figure 2.2 after they are mapped to this new feature space. Now, the vectors are linearly separable, and it is easy to find the optimal solution.

It turns out that SVMs do not actually have to map vectors to the new feature space. Instead, they use kernel functions. A kernel function takes two vectors in the input space and computes the dot product that would be obtained if these two vectors were first mapped to some other space. This is important for SVMs because the dot product is actually used to determine on which side of the optimal hyperplane a vector falls. By bypassing the need to actually map vectors to the new feature space, kernel functions allow SVMs to be computationally plausible. In fact, the choice of form for the kernel function that an SVM system uses is crucial to its performance, since this really determines the feature space to which document vectors are mapped, and only if the vectors are nearly linearly separable in this feature space is the system able to find a hyperplane that is useful for predicting whether or not future documents belong to the category being considered.

The discussion of SVMs so far has described how systems using this approach can handle a binary categorization task in which a separate YES/NO decision is made for every category. There are several ways in which the approach can be used, instead, for mutually exclusive categories. One approach is to compare every category with every other category, and to use these comparisons to order the likelihood of categories. (Remember that any binary categorization system can also be used when there are exactly two mutually exclusive categories, as described in Section 2.1). Another method is to use the distance between a document and the optimal hyperplane (determined using the kernel function) as a measure of the confidence that the document is, or is not, in each category. The document can then be assigned to the category with the highest confidence (or, if there is no category in which it is placed according to binary decisions, the category with the least confidence in the negative direction).

SVMs have many theoretical properties that researchers admire. A discussion of these properties is beyond the scope of this chapter, but they are discussed in (Cristianini and Shawe-Taylor, 2000) and (Burges, 1998), the two texts I recommended at the beginning of this subsection. Very briefly, there is a specific bound on the expected error rate that an SVM based system will achieve, given that it is using a kernel function that maps vectors to a space in which they are nearly linearly separable. In practice, various forms of kernels have tended to work well for specific tasks. See (Joachims, 1998) for a further discussion of why it is expected that SVMs will work well for text categorization.

In the text categorization literature, SVMs have been very successful for most of the tasks to which they have been applied. At the beginning of this subsection, I cite several sources that discuss the use of SVMs for text categorization tasks, and all have found that the approach has performed well. See, in particular, the work by Yang and Liu (1999), in which the system SVM^{light} (discussed in Section 2.8) beats all other systems tested for a formal comparison of systems based on experiments involving the Reuters data set (also discussed in Section 2.8). In my own work, however, I have not seen SVMs perform as well for my own text categorization tasks. Section 2.9 discusses many of the reasons that systems may perform very well for some text categorization tasks and not others. For now, I will just say that I suspect that SVMs tend to do better for binary categorization tasks, since that is what they are really intended for in principle, and my work has involved tasks involving mutually exclusive categories. Although I have discussed methods allowing SVMs to be used for mutually exclusive categories a few paragraphs back, I believe that they have drawbacks. It is also possible that the specific implementation of SVMs I have used - that which is provided as part of the Rainbow package described in Section 2.8 - is not as good as some other implementations. However, for one of my data sets from the corpus I have created (described in Section 3.1), I have also been able to test SVM^{light}, which is the exact implementation that is shown to be so successful by Yang and Liu (1999). This is intrinsically a binary categorization system, but it can be used for one of my data sets involving exactly two mutually exclusive categories. (This task involves the categorization of images as either *Indoor* or *Outdoor*; the discussion in Section 2.1 explains how a binary system can be used.) SVM^{light} performs significantly better than Rainbow's SVM implementation for this data set, but still not as well as Rainbow's Probabilistic Indexing system (this approach is discussed in Section 2.6.2.1) or as well as two of my own systems described in Chapters 4 and 5.

2.6.3 Some Other Approaches

There are many approaches that have been applied to text categorization, only some of which have been discussed in this chapter. I have chosen these techniques because they are among the more common approaches applied by researchers, and because they are the approaches used by the systems that comprise the publicly available Rainbow package (described in Section 2.8). Here, I provide a very brief description of some of the other commonly used approaches. Systems using decision list and decision tree approaches towards text categorization come up with a chain or tree of rules that, when followed, lead to a specific category (Cohen, 1995a; Quinlan, 1986; Quinlan, 1993). A Neural Network approach towards text categorization (Schutze, Hull, and Pederson, 1995) involves a network of units with inputs and outputs. The inputs to the first layer represent weights of terms, and the outputs of the final layer represent scores for categories. In between, edges have weights that propagate forward as nodes are activated. The learning phase consists of a technique such as backpropagation in which weights of edges are learned based on training examples. The Linear Least-Squares Fit (LLSF) technique is an example of a regression method that computes a $|C|$ by $|T|$ matrix (where C is the set of possible categories and T is the set of possible terms); this matrix is learned based on the training set, and by multiplying the matrix with the vector representation of a document, the system obtains a score for every possible category (Yang and Chute, 1994). The boosting approach towards text categorization (Schapire, Singer, and Singhal, 1998; Schapire and Singer, 2000) involves combining many *weak hypotheses*, each of which is a simple, moderately accurate hypothesis, sequentially in such a manner as to hopefully improve the accuracy of categorization at each step. Linear text classifiers produce category scores by taking the dot product (or cosine) of a document with vectors associated with each category. The Rocchio/TF*IDF approach discussed in Section 2.6.1.1 is the simplest and most common technique of learning such a classifier, but more complex approaches, such as the Widrow-Hoff and Exponentiated Gradient algorithms, are discussed in (Lewis et al., 1996).

2.7 Evaluation Metrics for Text Categorization

In order to compare various text categorization systems to each other, or to predict a single system's future performance for a given task, some method of evaluating each system's performance for the task must be employed. For a fair comparison, each system should be applied to the same data set, and the same metric should be used to evaluate each system. If researchers want to compare their results against previously published results, they must therefore make sure they apply their system to the same exact data set used for the published work, and then use the same metric used in published papers. An alternative way that researchers can compare their own system against a system discussed in the literature, if that system is obtainable, is for the researchers to run their own system and the competing system on their own data. To be fair, in this case, the evaluation metric that is used to compare systems should be chosen ahead of time, as it is possible that one metric may favor one system while some other metric favors another. Some of the issues involved with evaluating and comparing text categorization systems are discussed in (Lewis, 1995).

The simplest of the commonly used metrics to evaluate systems is probably overall accuracy, i.e. the percentage of a system's predictions that are correct. For tasks involving mutually exclusive categories in which there is not a single category that dominates, this is often a very good measure of performance, and it is one that I often use throughout this thesis. Since every document must be assigned to exactly one category, a system makes only one prediction per document. If there are n documents in a test set and k are assigned to the correct category by a system, the accuracy of the system is k/n. (Often this fraction is multiplied by 100 and expressed as a percentage.) Another equivalent performance metric is overall error, which is simply one minus overall accuracy; this represents that percentage of a system's predictions that are wrong.

Overall accuracy and overall error count every decision as entirely correct or entirely incorrect. In general, this seems fair, and is accepted by most researchers. In Section 8 of (Sable and Hatzivassiloglou, 2000), a co-author and I toy with an alternative metric that allows predictions to be weighted as partially right or wrong, depending on confidence. For example, a high-confidence, correct prediction is rewarded more than a low-confidence, correct prediction; a high-confidence, incorrect prediction is likewise penalized more than a low-confidence, incorrect prediction. However, this type of evaluation is quite rare, and I do not discuss it further in this chapter or this thesis.

Overall accuracy is not an appropriate evaluation metric for binary categorization tasks, especially when most documents belong to very few of many possible categories. As described in Section 2.1, binary categorization tasks require a separate YES/NO decision for every document/category pair, and a system that decides NO for every decision can have a very high overall accuracy, but clearly such a system is trivial and not useful. The same problem can occur when dealing with mutually exclusive tasks if one category clearly dominates the others in abundance; a system that assigns every document to this largest category may achieve a high overall accuracy.

Partly to deal with this problem, alternative measures have been developed that are indicative of a system's performance for individual categories. The two most common measures are precision and recall. Precision represents the percentage of documents assigned by the system to a category that actually belong to that category. Recall represents the percentage of documents actually belonging to a category that are assigned to that category by the system. A third measure that sometimes appears in the text categorization literature is fallout, which represents the percentage of documents not belonging to a category that are mistakenly assigned by the system to the category. This is considered an alternative to precision; it seems to be less common and, in my opinion, it is less intuitive, so I do not discuss it further in this chapter.

More formally, let Table 2.2 be the *contingency table* for some single category of interest based on the results of a system that has been applied to a data set. TP represents the number of *true positives*, which is the number of documents that

	Actual YES	Actual NO
System YES	TP	FP
System NO	FN	TN

Table 2.2: This is a generic contingency table for a single category.

are assigned by the system to the category and actually belong to the category. FP represents the number of *false positives*, which is the number of documents that are assigned to the category but do not actually belong to the category. (For the case of a false positive, if we are dealing with binary categories, this does not imply anything about the other categories; if we are dealing with mutually exclusive categories, this implies that the documents belong to one of the other categories.) FN represents the number of *false negatives*, which is the number of documents that are not assigned to the category but should have been assigned to the category (i.e. they actually belong to the category). Finally, TN represents the number of *true negatives*, which is the number of documents that are correctly not assigned to the category (i.e. they do not belong to the category). (For the case of a true negative, if we are dealing with mutually exclusive categories, it does not matter if these documents are assigned to the correct category; for example, if Table 2.2 represents the contingency table for some category c_1, and a given document is assigned by the system to some other category c_2 when it really belongs to a third category c_3, this incorrect prediction still increases the count of TN for the contingency table of c_1.)

Based on this contingency table for a single category, the precision and recall of the system for the category are:

$$P = \frac{TP}{TP + FP},\ R = \frac{TP}{TP + FN} \tag{2.18}$$

In other words, the precision is the number of documents correctly assigned to the category divided by the total number of documents assigned to the category, while recall is the number of documents correctly assigned to the category divided by the number of documents belonging to the category. (If the formula for either precision or recall results in a denominator of zero, the corresponding measure is considered undefined.) A perfect system would achieve a precision and recall of 1 (or 100%).

In actuality, no text categorization system is perfect. Some documents that do not belong in a category are sometimes assigned to the category, and some documents that do belong in the category are not assigned to the category. Typically, a system can be tweaked to improve one measure for a category at the expense of the other. If parameters are adjusted so that more documents are assigned to a particular category, the recall for that category likely improves but the precision likely gets worse. On the other hand, if parameters are adjusted so that less documents are assigned to the category, the recall likely goes down but the precision likely improves. At times, one of these two measures may be more important than the other. For example, in an application that filters out spam from incoming e-mail, the precision for the category *spam* is probably more important than recall, because allowing a piece of spam to fall through is probably not as bad as filtering out an important message.

To compare systems to each other, it is nice to have a single metric representing performance. If both precision and recall are being used, for example, it is difficult to compare two systems to each other if one has better precision but the other has better recall. Several techniques therefore exist that combine precision and recall into a single measurement. A good analysis of these techniques is provided in (Yang, 1999), and I summarize them here.

One such technique is to use the *eleven-point average precision*. This takes advantage of the fact that parameters can generally be tweaked to increase either precision or recall at the expense of the other. The idea here is to tweak the values repeatedly in order to achieve recall values of 0.0, 0.1, 0.2, ..., 0.9, and 1.0. At each of these recall levels, precision is measured, and the 11 obtained precisions are averaged together into a single measurement.

Another technique is to use the *breakeven point* (BEP), which is the value at which precision and recall are equal to each other. Like the eleven-point average precision, obtaining this value entails tweaking parameters of a system. If it is impossible to achieve values of precision and recall that are exactly equal, then the average of the nearest precision and recall can serve as an *interpolated BEP*.

A third technique to combine precision and recall with a single metric is the F measure (van Rijsbergen, 1979). Formally:

$$F_\beta = \frac{(\beta^2 + 1) \times P \times R}{\beta^2 \times P + R} \tag{2.19}$$

Here, β determines how to weight precision versus recall. If $beta = 0$, then $F_\beta = P$, and as $\beta \to \infty$, $F_\beta \to R$. As mentioned a few paragraphs back, for certain tasks, one of either precision or recall may be considered more important than the other, and in these cases, the F_β metric certainly offers an important advantage. Typically, though, precision and recall are weighted equally, in which case we are dealing with the F_1 measure, a special case of the F_β measure defined as follows:

$$F_1 = \frac{2 \times P \times R}{P + R} \tag{2.20}$$

The F_1 measure is always closer to the lower of precision and recall, and therefore requires a good score for both of these metrics in order to indicate good performance for a category.

As pointed out in (Yang, 1999), the BEP for a system applied to a task is always less than or equal to the optimal F_1. This is because the BEP is really an instance of the F_1, at that point where precision and recall are equal. This is because, at that point, we have:

$$F_1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times P \times P}{P + P} = \frac{2 \times P^2}{2 \times P} = P = R = BEP \tag{2.21}$$

So we know that the F_1 measure is equal to the BEP if the settings of parameters are just right, and it may, in fact, be higher (or lower) at other settings. In any case, this is an example of why it is not fair to compare systems using similar but different metrics, even if they are applied to exactly the same task.

In my own work, I have relied mainly on the F_1 to measure performance for individual categories. There are many advantages of this metric. The most important advantage is that it does not require a system to be tweaked over and over again to obtain various precision/recall results for each category. This does not really make sense, especially when dealing with mutually exclusive categories. An actual system will, in practice, only be applied once to a data set. The F_1 can be used to measure performance for all categories based on this single run. Throughout this thesis, I typically provide F_1 measures for all categories, as well as overall accuracy measures of systems. Since I typically deal with mutually exclusive categories in which no single category dominates, overall accuracy is the measurement I rely on the most to compare systems.

As I have explained earlier in this section, when dealing with binary text categorization tasks in which most documents do not belong to most categories, overall accuracy is not a reliable measure of performance for a system. This is the case, for example, when dealing with the Reuters data set, described in Section 2.8. Precision and recall are still applicable, and can be combined with a single metric such as the F_1 measure for each individual category. However, it is still convenient to have a single measure of performance based on the entire data set (and all categories) in order to compare systems. For this reason, it is common to average precision, recall, and especially F_1 measures over all categories. This can either be done treating all documents as equal (micro-averaging) or treating all categories as equal (macro-averaging). Micro-averaging creates a single, global contingency table in which the value of each cell is the sum of the values of the corresponding cells from the per-category contingency tables; the micro-averaged metrics are then computed based on this global contingency table. Macro-averaging computes the metrics for each individual category first and then averages these scores together to compute the macro-averaged metrics. In the text categorization literature, the micro-averaged F_1 seems to be more commonly used than the macro-averaged F_1 to compare systems (Yang and Liu, 1999). In this thesis, when I deal with binary categories (e.g. at the end of Chapter 4 when I test my density estimation system on the Reuters data set described in Section 2.8), I report both of these measures.

2.8 Common Corpora and Publicly Available Systems

As has been mentioned in the last section, there are two ways that a researcher can compare his system to others. One is to test the system on the exact same data set as that of other researchers who have already published results. To aid this purpose, there are a few publicly available text categorization corpora that are commonly used for published papers. The other method is for the researchers to test their own system and the competing systems on their own data sets. This is only possible if the competing systems are made available; luckily, a few such publicly available text categorization systems exist.

The most commonly used text categorization corpus is the Reuters corpus (Lewis, 1997), which has already been mentioned several times throughout this chapter. The current version of this collection is Reuters-21578, and this is currently available on the web at http://www.daviddlewis.com/resources/testcollections/reuters21578 or http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html. The first of these two sites is maintained by the David Lewis, the creator of the corpus, and from here there is also a link to a site at which a newer, supposedly better version of the corpus will soon be made available. The Reuters-21578 collection comes with several *splits* that determine exactly what portions of the data should be used with training and testing. Both of the collection sites contain a link to a README file (Lewis, 1997) that describes the corpus and the various possible splits.

The split that I have used for my own work is referred to as the ModApte split of the Reuters-21578 corpus. This split includes 9,603 training documents, 3,299 test documents, and 135 categories. The categories are described as being "economic subject categories" (Lewis, 1997) such as *coconut*, *gold*, *inventories*, and *money-supply*. These are binary categories, so it is possible for a document to be assigned to no categories, one category, or multiple categories. Two of the many published papers comparing the performance of multiple systems on this data set are (Yang, 1999) and (Yang and Liu, 1999). I consider both of these papers impressive for their clear descriptions of procedure, their exploration of systems using many different approaches, and their evaluation using several metrics.

To be able to directly compare the results of my systems tested on this data set with the published results of Yang and Liu (1999), I have applied the same pre-processing to the corpus, as have some other researchers (e.g. (Joachims, 1998)). This consists of first eliminating all categories that do not contain at least one training document and one test document and then disregarding all unlabeled documents. This process leaves 7,770 training documents, 3,019 test documents, and 90 categories. Most of the remaining documents documents (9,160 of the 10,789, or 84.9%) are assigned to exactly one category, and the rest are assigned to more (since the pre-processing disregarded documents without any categories). The largest number of categories assigned to any single document is 15, and the average number of categories assigned to a document is 1.24. The category distribution is quite skewed. The most common category has 3,964 instances (2,877 training instances and 1,087 test instances), while, on average, each category has only 148 instances. Some categories have just one training instance and one test instance.

Another publicly available corpus sometimes used for text categorization research is the OHSUMED test collection (Hersh et al., 1994), which is actually a subset of the MEDLINE database. This collection is available via ftp from ftp://medir.ohsu.edu/pub/ohsumed, along with a corresponding README file. The documents are abstracts with titles from medical journals, and the categories are MeSH (Medical Subject Heading) indexing terms. As is Reuters, OHSUMED is a binary text categorization corpus. Research applying text categorization systems to this corpus includes (Lewis et al., 1996), (Yang, 1997), and (Joachims, 1998).

A third corpus that has commonly been used to test text categorization systems is the TREC-AP test collection. In this collection, the documents (of which there are hundreds of thousands) are AP headlines and the categories are relevant topics such as *federal budget* and *Nielsons ratings*. As are Reuters and OHSUMED, the AP test collection is a binary text categorization corpus. Much of the text categorization research using this data set (Cohen, 1995b; Lewis et al., 1996; Schapire and Singer, 2000) uses ten categories originally defined by Lewis and Gale (1994).

A fourth commonly used corpus for text categorization research is the 20 Newsgroups collection collected by Lang (1995). This data set is available on the web at http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html or http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html. The documents are messages that have been posted to 20 Usenet newsgroups (1,000 from each), and the categories are the newsgroups themselves. Although many text categorization researchers have treated these categories as mutually exclusive (Joachims, 1997; McCallum and Nigam, 1998; Nigam, Lafferty, and McCallum, 1999; Nigam et al., 2000), Schapire and Singer (2000) point out that some articles are posted to multiple newsgroups, and they argue that this should be treated as a binary text categorization corpus.

A fifth commonly used text categorization corpus is the WebKB dataset (Craven et al., 1998). This data set is available on the web at http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data. The data set consists of over 8000 web pages; about half were taken from the computer science departments of four universities, and the rest were taken from miscellaneous departments of other universities. The categories are *student*, *faculty*, *staff*, *department*, *course*, *project*, and *other*, and they are mutually exclusive. Research applying text categorization to this corpus includes (McCallum and Nigam, 1998), (Blum and Mitchell, 1998), (Nigam, Lafferty, and McCallum, 1999), and (Nigam et al., 2000).

The corpora mentioned so far in this Section allows researchers to compare their systems to others with published results. Researchers must make certain to use the same data set with the same breakdown of training and test documents, and the same evaluation metrics must be used. The other way that researchers can compare competing systems to their own is to run the competing systems on their own data. This is only possible if the competing systems are obtainable. Luckily, there are a few publicly available text categorization packages that give access to systems relying on a variety of approaches.

One publicly available text categorization package is the *Rainbow* package (McCallum, 1996), available on the web at http://www-2.cs.cmu.edu/˜mccallum/bow/rainbow. This package is based on the *Bow* library (McCallum, 1996), available on the web at http://www-2.cs.cmu.edu/˜mccallum/bow, which also includes front-ends for clustering and information retrieval. The Rainbow package is comprised of several text categorization systems relying on a variety of approaches. These approaches are Rocchio/TF*IDF, K-Nearest Neighbor, Naive Bayes, Probabilistic Indexing, and Support Vector Machines; in other words, all of the approaches explained in Sections 2.6.1 and 2.6.2. The systems that comprise the Rainbow package assume that categories are mutually exclusive, and therefore they have been appropriate for my own research which has concentrated on tasks involving such categories. Throughout this thesis, I compare performance of my own systems to performance of the systems that comprise the Rainbow package.

Another publicly available categorization system is SVM^{light}. This system is available on the web at http://svmlight.joachims.org. Strictly speaking, this is not just a text categorization system; rather, it is a categorization system that can accept, as input, feature vectors representing documents such that each feature has a real value. SVM^{light} can thus be used for text categorization by defining the features for documents to be the set of possible words (or terms, or stemmed words, etc.). SVM^{light} has been extremely successful, as discussed in the text categorization literature, particularly for the Reuters data set (e.g. see (Yang and Liu, 1999), in which SVM^{light} outperforms all competing systems). SVM^{light} assumes binary categories, and so it is not appropriate for most of the tasks I have focused on in my own research. (There are ways to convert binary decisions into decisions for mutually exclusive categories, but I believe they have drawbacks, and I have not applied them using this system.) I have tested SVM^{light} on a task involving the categorization of images as *Indoor* or *Outdoor*. As explained in Section 2.1, systems intended for binary text categorization tasks can be applied to tasks involving exactly two mutually exclusive categories. For this task, SVM^{light} outperforms the implementation of SVMs included with the Rainbow package (discussed in the previous paragraph), but it does not do as well as some other systems, including my own density estimation system discussed in Chapter 4 or my BINS system discussed in Chapter 5.

As explained earlier in the section, public corpora and systems allow researchers to compare their techniques against others. The contributions of my own work include the creation of a new text categorization corpus and a new text categorization system. The corpus is described in detail in Section 3.1. It is unique among publicly available text categorization corpora in that it includes images with labels and associated text. It also includes four separate sets of categories, two of which are hierarchical. Instructions for obtaining this corpus will soon be posted at http://www.cs.columbia.edu/~sable/research/corpus.html. The system is BINS, described in detail in Chapter 5. The approach this system uses is a generalization of Naive Bayes (explained in Section 2.6.1.3) in which term weights are estimated for groups of words that share statistical features in common; this can be thought of as a smoothing technique that avoids inaccurate term weights for individual words with scarce evidence. The system is user friendly with well-documented code and an interface similar to that of the Rainbow package discussed earlier in this section. Instructions for obtaining this system will soon be posted at http://www.cs.columbia.edu/~sable/bins.html.

2.9 Various Properties of Data and Categories

This chapter has discussed many approaches for text categorization. Some, such as those discussed in Section 2.6.1 (Rocchio/TF*IDF, K-Nearest Neighbor, and Naive Bayes), are based on simple, intuitive principles, while others, such as those discussed in Section 2.6.2 (Probabilistic Indexing, Maximum Entropy and Support Vector Machines), are more advanced. These approaches have all commonly been applied to text categorization tasks in the literature without a clear winner. Most of the literature seems to suggest that the advanced approaches are better than the basic ones; however, this has been based on experiments with a limited number of corpora - primarily those discussed in Section 2.8, with the Reuters corpus easily being the most commonly used corpus - and, in my own research experience, I have not seen this to be the case. With some improvements, I show in this thesis that modifications of the basic techniques are quite competitive. For example, the addition of density estimation to a Rocchio/TF*IDF approach (discussed in Chapter 4) or the smoothing of a Naive Bayes approach using bins (discussed in Chapter 5) lead to systems that meet, and sometimes beat, the performance of all other systems tested for tasks involving data sets of a corpus that I have created (discussed in Section 3.1). In this Section, I argue why I believe that there is no clear winner in the field of text categorization. The gist of the argument is that the properties of the categories and the data used for text categorization tasks are highly variable, and which approaches are likely to do well for a given text categorization task depends on these properties. I make an analogy to the "No Free Lunch" theorems (Wolpert and Macready, 1995; Wolpert and Macready, 1997), and argue that no method should be expected to perform the best for all cases.

The properties of categories involved with text categorization tasks can vary highly. Categories can be mutually exclusive or independent; categories can be general or specific; categories can be nominal or action oriented; there might be many categories or there might be few, and categories may have approximately equal representation or the distribution might be very skewed. These properties of categories can all potentially affect the likelihood that one type of system (i.e. a system that uses a particular approach) outperforms another. The next few paragraphs expand what I mean by each of these properties and how this might affect some of the approaches that have been discussed in this chapter.

As first discussed in Section 2.1, categories are sometimes mutually exclusive and exhaustive, in which case inclusion of a document in one category excludes the inclusion of the document in all other categories; in other cases, categories are binary, in which case the relationship between categories is arbitrary, and most systems then treat the categories as independent. Some approaches are really set up, in principle, for one of these two cases, and can only be applied to the other case after certain hacks are applied. For example, the Rocchio/TF*IDF approach computes a similarity between a document and every possible category. If the categories are mutually exclusive, a system using such an approach can assign the document to the category with the highest similarity score. However, if the categories are binary, some additional step is needed to convert the similarity scores to YES/NO decisions for every category. A few of the possible methods of doing this are described at the end of Section 2.6.1.1. All are a bit ad-hoc, and this might explain why Rocchio/TF*IDF seems to underperform other approaches in much of the text categorization literature (e.g. see the comparison of many approaches by Yang (1999)), which generally focuses on binary text categorization tasks. In my own work, however, which focuses on mutually exclusive categories, I have seen that Rocchio/TF*IDF often performs well, especially when aided by density estimation as explained in Chapter 4. On the other hand, consider an SVM approach towards text categorization, which computes a hyperplane for every category. For binary categorization, a document can be placed in or excluded from a category depending on which side of the hyperplane the document falls. However, if the categories are mutually exclusive, some additional step is necessary in order to determine to which single category a document should be assigned. A couple of methods of doing this are mentioned in Section 2.6.2.3, but again, they are somewhat ad-hoc. For example, one approach that is used is to consider the distance of the document from a hyperplane as a measure of confidence for the corresponding category, but there is no real reason to believe that different categories should share the same scale. This might explain why SVMs seem to perform so well in the literature (e.g. see (Yang and Liu, 1999)), but they do not perform extraordinarily well for many of the tasks I consider in my own research.

Text categorization has many purposes, some of which are mentioned in Section 2.2. Some of the potential applications deal with categories that are quite general; for example, the categorization of e-mail as *spam* versus *not spam*. The category *not spam* is especially vague, representing any type of e-mail that the user would not want filtered, and while *spam* might seem specific at first, most people who commonly use e-mail probably realize that this comes in many forms. Other uses of text categorization deal with categories that are more specific. In my own research, I deal mainly with categories of my own creation that apply to news documents and their embedded images (the corpus I have created is discussed in Section 3.1); this type of categorization can aid browsing and search capabilities, as is shown in Chapter 9 which discusses Columbia University's Newsblaster system. The sets of categories that apply to my corpus range from the general (*Indoor* versus *Outdoor*, applying to images) to the specific (*Workers Responding*, *Affected People*, *Wreckage*, or *Other*, applying only to images embedded in news documents about disasters). The multiple sets of categories that apply to my corpus also represent different levels of abstraction; one is topical and deals with content (*Struggle*, *Politics*, *Disaster*, *Crime*, or *Other*, applying to full news documents), while another deals with the general setting of an image (*Indoor* versus *Outdoor*). Some text categorization corpora contain categories that are even more specific than any of the ones mentioned so far; for example, the commonly used Reuters data set discussed in Section 2.8 contains categories such as *potato* and *income*.

Although it might seem like there is no obvious reason that the distinctions mentioned in the previous paragraph should affect which systems perform well, there are reasons that this is likely the case. For one, the specificity versus generality of the categories affects the data. In other words, if categories are very specific - e.g. *potato*, which is literally a Reuters category applying only to economic articles about potatoes - the documents within a category will likely have very similar text, and this helps certain approaches more than others. On the other hand, if the categories are vague and abstract, such as the *Indoor* and *Outdoor* categories, documents within a category will have a much greater variance. Later in the section, I discuss why this issue involving training data influences various approaches to different extents. Another reason that the specificity or generality of categories can affect different systems is that it might affect what types of words tend to be important. I am thinking here of the authorship attribution task considered by Mosteller and Wallace (1963; 1964). Here, the categories are the possible authors, and these are clearly general in the sense that any author may choose to write about any topic; it turns out that for such categories, content words are not important at all, whereas filler words such as "an", "of", and "upon" are quite indicative. A Rocchio/TF*IDF approach has no chance with these categories, since these words are automatically given very low weight. Other approaches discussed in this chapter also have no chance if such words are filtered out, but variations of approaches such as Naive Bayes that do not filter out words and consider counts as well as presence may do well (this is basically the approach used by Mosteller and Wallace).

Some categories are nominal, and what I mean by this is that the mere mention of a single object or concept in a document might by enough to determine a category. For example, one set of categories defined for my own corpus (described in Section 3.1) deals with types of events discussed in news documents, and the word "earthquake" is a great indicator that the document belongs in the *Disaster* category. However, other categories involve determining the focus or emphasis of a document. For example, another set of categories that applies to the images embedded in the *Disaster* documents deals with the focus of the images and the actions taking place. A word such as "victim" in a caption is a good indicator of the *Affected People* category if it is the main subject of the caption, but if the "victim" is being helped by a "rescuer", and "rescuer" is the main subject, the image is likely more appropriate for the *Workers Responding* category. This is made more clear in Chapter 6, in which I argue that all bag of words approaches are likely to have trouble with tasks such as this (i.e. those relying on focus, perspective and point-of-view). However, I describe another system in that chapter, one that relies on deeper linguistic processing, that performs better for this specific task.

Finally, it is plausible that certain approaches may fair relatively better when dealing with just a few categories while others may fair relatively better when dealing with many categories. If the categories are binary, in which case most systems consider them to be independent, this probably doesn't make any difference, but if the categories are mutually exclusive, it might. It is quite possible that some approaches may fair relatively better if categories are approximately equally distributed, while others may fair relatively better when the distribution is more skewed; and one might expect that a skewed distribution is more likely when there are many categories. For example, if there are many categories with a skewed distribution, a kNN system may favor large categories and penalize small categories to an unfair degree. It is more likely that the majority of *neighbors* of a new document belong to the larger category just because there are more of these documents in the training set. While it is good to favor larger categories to some degree, the kNN system may give them an unfair advantage. A Naive Bayes system, on the other hand, may not give larger categories enough of an advantage. Although the skew affects the estimated *a-priori* probabilities of categories, it is rare that this makes any difference for a prediction because the final probabilities of a Naive Bayes system are usually exaggerated (as explained in Section 2.6.1.3).

In addition to variations in properties of categories, there are several properties of data that vary from one categorization task to another. The training set can be large or small (i.e. there may be many training documents or there may be few); the individual training documents might be short or they might be long; documents within a category may tend to be very similar to each other, or there may be lots of variety within each category; the labels of the training documents might be highly accurate or there might be a significant degree of error. These properties of data can all potentially affect the likelihood that one type of system (i.e. a system that uses a particular approach) outperforms another. The next few paragraphs expand what I mean by each of these properties and how this might affect some of the approaches that have been discussed in this chapter. Some of these properties may be influenced by the specific task being considered (i.e. the set of categories may indirectly affect some of these properties), while others depend only on the training data obtained.

All other things being equal, it is expected that all machine learning systems will perform better with more training data than with less. In other words, as more and more accurate and representative training examples are added to a training set, the performance of any system is likely to improve (perhaps with diminishing returns as some optimal performance is approached). However, some systems are affected more than others by the need for lots of training data. If curves representing performance versus training set size are plotted, they will not have the same slopes at various points for various systems using distinct approaches. It is therefore possible that the system that performs the best when there is little training data may not be the system that performs the best when there is lots of training data. Although I can not claim to have a clear intuition as to which systems are more affected by training set size than others, I have certainly noticed evidence in my own research experience that this is the case. For example, I show in Section 4.3.3, that when applied to the categorization of binary Reuters categories, a system using density estimation outperforms a system using a standard Rocchio/TF*IDF approach for large categories, but the opposite is true for small categories.

Depending on the task being considered, documents used for text categorization tasks can vary highly in length. News articles may tend to contain more text than e-mails or web pages, and image captions have a lot less text than any of those. The formality of the text also depends on the type of documents being considered. For example, e-mails tend to be more casual than news articles, and this might mean that they vary more. Once again, it is not clear how these issues affect various systems, but it probably does not affect all approaches the same way. Image captions probably do not contain important terms more than once, whereas news articles may contain important terms many times. We have already seen that different approaches handle term frequencies differently; for example, many Naive Bayes implementation only use binary weights for terms, therefore not taking this into account at all, whereas a Rocchio/TF*IDF approach uses statistical weights that do take term frequencies into account. It is also possible that approaches that use deeper NLP processing (such as that described in Chapter 6) may have a better chance when documents are small; it is easier to parse a single sentence, for example, than an entire article, and when there is little text, it might also be more important to determine additional information such as the main subject or action taking place.

One issue involving data that is influenced by the categories being considered is whether or not the documents within a category tend to all be similar to each other, or whether there is a lot of variety within a category. For example, within the Reuters category *potato* (applying only to economic articles about potatoes), it is likely that all the documents are very similar to each other, because the category is extremely specific. In a category such as *Politics*, this is less the case, although there are certain terms and concepts that might be included in most documents. A category such as *Outdoor* likely consists of a huge variety of documents that do not share much in common at all. This property clearly affects certain approaches more than others. For example, a Rocchio/TF*IDF approach is really computing a centroid for every category, and comparing new test documents to all of the centroids. Centroids are not very reliable for categories with a lot of variety. A kNN approach, on the other hand, is not affected by this, as long as there are some very similar training documents with the same category as the new document being considered (i.e. the existence of other, non-similar training documents with the same category does not hurt). And for a Naive Bayes approach, it is not really important that any training documents are very similar to the new document being considered, as long as the individual words in the new document are more likely in the correct category. This certainly does not mean that Naive Bayes is always better; there may be tasks for which any individual word might be found in any category, but looking at a document as a whole is more indicative. In any case, this is an issue that affects different approaches to different extents.

Finally, depending on how training data is obtained, some training sets may be more accurate than others. If labels are taken from many experts, and only training documents for which there is agreement are used, it should be expected that the labels are more accurate than if, for example, only one non-expert is used. Labels may be even less accurate for a training set that has been created in some automatic fashion without human intervention. All other things being equal, all systems are expected to perform better if the training data is more accurate, but the level to which different approaches are affected by inaccuracies in the training set is not necessarily the same for all cases. For example, in Appendix P, I compare the use of an automatically generated training set for Newsblaster (described in Chapter 9) to one using manually labeled documents. Although the automatically generated training set is larger, it is also less accurate, and all tested systems perform better with the manually created training set. Still, the degree to which each system is affected varies highly. The system least affected performs only 3.1% worse when the automatic training set is used, while the system most affected performs over 40% worse.

We have thus seen that there are many properties associated with text categorization tasks that can vary. We have also seen that many approaches to text categorization exist, and I have already expressed that there is no clear winner in the field. I believe that these two facts are related. The text categorization literature seems to support the idea that recent, advanced approaches should be expected to win out over basic approaches. In my own experience, however, I have not seen this to be the case. Most published papers on text categorization base results on just a few public corpora, and a very large proportion of these use the Reuters corpus. This corpus consists of many specific, binary categories.

I believe that the reason that there is no clear winner for all text categorization tasks is related to the so-called "No Free Lunch" (NFL) theorems (Wolpert and Macready, 1995; Wolpert and Macready, 1997). These theorems state that any two algorithms used to search the same space perform the same over all possible cost functions. In other words, for every case where one such algorithm performs better than a second, there is some other case where the second algorithm performs better than the first. There has been work that shows that NFL also applies to tasks such as cross validation (Zhu and Rohwer, 1996). This does not mean that methodologies such as cross validation are not useful. What it does mean is that such methodologies are making assumptions about the expected data, and they will only perform well if the assumptions are true. The NFL theorems, as stated, do not directly apply to text categorization. Still, I believe that the general result still holds; that is, the approach that works the best for a specific task is the one that best fits the data, and certain properties of the data are often indirectly determined by the specific categories.

Of course, it would be wonderful if one could determine which approaches are likely to work well for tasks with specific properties without having to test all possible approaches. This is beyond the scope of this chapter and thesis; in fact, I believe it is a very difficult task that would necessarily involve a lot of experimentation with many systems and many corpora containing data and categories that exhibit various properties. This could be worthy of an entire thesis in and of itself. For now, it is necessary to perform cross validation experiments or experiments with a tuning set (as explained in Section 2.5) in order to determine which approaches perform well for the task at hand.

2.10 Concluding Discussion of Text Categorization

In this chapter, I have introduced the reader to the field of text categorization. I have formally defined the task and listed some of the potential applications. I have described the essence of the *bag of words* approach used by almost all modern systems to represent documents. I have explained that machine learning approaches learn a classifier from a training set, often tuning parameters with a technique such as cross validation. I have described many of the common approaches used for text categorization, including a few basic approaches and a few that are more advanced, and I have explained some of the metrics used to evaluate them. Finally, I have discussed many of the properties of categories and data that vary highly from one task to another, and I have provided some insight into how these properties may determine which system performs the best for a given task.

There is so much future work to be done in the field of text categorization. Although there have been some large studies comparing approaches (e.g. (Yang, 1999) and (Yang and Liu, 1999)), such studies have generally relied on one or a few publicly available corpora. A more general study is needed that compares different systems for many possible tasks. It would be nice if one could come up with some sort of rules to determine which approaches would likely work the best for which types of text categorization tasks. For now, it is probably best to rely on cross validation or a similar technique to determine which approach is the best for a given task.

As discussed in Section 1.3, text categorization research generally fits into one of two paradigms - the exploration of techniques or the exploration of representation. Most of the recent literature fits into the first of these two paradigms; that is, researchers have been attempting to create better machine learning techniques. Some of my own research fits into this paradigm as well. I describe in Chapters 4 and 5, for example, two new machine learning approaches that I have developed. However, it is not clear that there isn't more room for improvement in the second paradigm, especially when looking at tasks that are slightly out of the ordinary.

In my own research, I have dealt mainly with images. I show in Chapter 3 (and throughout this thesis) that the categories and the text associated with images differ substantially from those associated with lengthier text-only documents such as articles or e-mails. As explained in Section 2.9, these differing properties may mean that the approaches which generally perform the best for more standard text categorization tasks may not perform the best for those that I am commonly dealing with. In addition, images allow for a representation that text-only documents do not - namely, a representation involving low-level image features. For all of these reasons, much of my own research falls into the second of the two paradigms discussed previously; that is, exploration of representation. In Chapter 6, for example, I discuss the creation of a system that does not rely on a bag of words representation, but instead uses deeper NLP techniques to determine the main subject and verb of an image caption; this system beats all competing systems tested for a specific task. In Chapter 8, I discuss the use of low-level image features, in addition to text, to categorize images.

Chapter 3

Categorizing Images Using Text

I have shown in Chapter 2 that many text categorization methodologies exist, and that virtually all of the popular methods today rely on machine learning approaches and bag of words representations for documents. In this chapter, I begin to discuss the use of such techniques to categorize images and the documents that contain them. For some sets of categories, the pre-existing approaches work fine, but for other sets of categories, this is not the case. As described in Section 1.3, the algorithms used to automatically label documents are important, but sometimes so is the representation of the documents themselves. When dealing with images, we have access to a type of information that has not been available for text categorization problems in the past; namely, image features. In addition, the properties of the text associated with images, and the properties of the categories likely to be associated with images, can differ substantially from those associated with full-length text documents such as articles, e-mails, or web pages. For example, if the text is an image caption, we are dealing with much less data per document, and categories are more likely to place on emphasis on the focus of the text. For these reasons, representation becomes more important when dealing with images. Later in the thesis I show that novel approaches relying on non-standard representations, involving image features or linguistic analysis, have led to substantial improvements for the categorization of images.

In this chapter, I discuss the creation of a text categorization corpus consisting of multimedia news documents with captioned images and corresponding articles. Multiple sets of categories representing various levels of abstraction are described. I show that for some sets of categories, existing text categorization systems achieve excellent results simply by using the associated text to categorize the images or entire documents. I show in Chapter 8 that, even in these cases, improvement can sometimes be gained by also considering image features. For other sets of categories, existing systems do not perform well. In Chapter 6, I provide an explanation for why this occurs and show that NLP techniques are necessary to achieve optimal results.

3.1 The Creation of a Text and Image Corpus

In order to explore the use of text categorization to automatically label images, I have created a corpus containing a substantial number of images with associated categories. To the best of my knowledge, there does not exist any other publicly available text categorization corpus containing images other than the one I have created. To create such a corpus, I have collected appropriate data, carefully defined relevant and interesting categories, and collected manual labels for the documents. I have decided to work in the news domain, with documents consisting of articles with embedded captioned images. These were readily available at the time when I was creating the corpus, I have a personal interest in the news domain, and the potential benefits of being able to classify such documents becomes clear in Chapter 9, which discusses Newsblaster, a system that showcases the work of Columbia University's NLP group and has already become quite popular on the web (a recent analysis indicates that Newsblaster receives tens of thousands of hits every day).

As described in Section 2.1, categories used for text categorization tasks are generally either binary (in which case a separate YES/NO decision is required for each category) or mutually exclusive (in which case each document belongs to exactly one category). I have chosen to define sets of mutually exclusive categories. I believe this type of categorization has been under-represented in the text categorization literature, largely because the most popular public text categorization corpus (Reuters) uses binary categories. In addition, I believe that mutually exclusive categories tend to be more appropriate when dealing with the news domain. Articles in newspapers or the on-line equivalent tend to be placed in a single section representing a general area of news. Also, I envision an eventual hierarchy of categories which would have the form of a tree like structure where each image or other document falls into exactly one of the nodes.

3.1.1 The Raw Data

The raw data used for the corpus consists of tens of thousands of news postings from a variety of Usenet newsgroups. The dates of the postings range from November 1995 through January 1999. All postings contain a text article and some contain one or two images with associated captions. For the final corpus I have created from this data, I consider a document to be an image along with its corresponding caption and article. There are 2,312 such documents in total. There are some instances of two separate images sharing the same article (in which case the article exists twice in my final corpus), or a single image appearing with two separate articles (in which case the image exists twice in the final corpus), but both of these cases are rare. Most of the image captions are two or three sentences long. Typically, the first sentence describes the image and the rest gives background information about the related story. Articles are generally several paragraphs long, with lengths typical of stories in ordinary newspapers.

I have written a script that parses the raw data and extracts, from each document, an image/caption/article triple; I assign the same identification number to each item in a triple. I also store articles that do not have associated images. These articles are assigned separate identification numbers; they can be used for certain work with unlabeled data (e.g. the work discussed in Appendix N), or for certain other purposes such as calculating IDF values. I also store image/caption pairs that do not have associated articles, again using separate identification numbers, although such cases are rare.

3.1.2 The Categories

I have defined multiple sets of categories for the images and documents in my corpus. These sets of categories represent different levels of abstraction, and each set of categories leads to its own data set for experimentation. For each data set, I have chosen categories such that each category represents a significant portion of the data without having any single category dominate. Categories from pre-existing text categorization corpora (e.g. the economic subject categories used for the Reuters corpus described in Section 2.8) are simply not appropriate. Based on my own inspection of the documents, I have created categories that apply to the news images in my corpus and the documents that contain them. I have attempted to choose categories that are intuitively understandable and interesting in such a way that users might want to narrow searches or browsings to a specific category. Further reasons and motivations for choosing these categories are discussed in later sections when I describe the categories in more detail; here I only briefly introduce the categories.

One set of categories applies only to images, and the categories are *Indoor* and *Outdoor*. The next data set involves broader, high-level topical categories that apply to entire documents (i.e. an image along with its associated caption and article), and the categories are *Struggle*, *Politics*, *Disaster*, *Crime*, and *Other*. In the remainder of this proposal, I refer to these categories as the Events categories. Another set of categories only applies to images that are part of the *Disaster* documents, and the categories are *Workers Responding*, *Affected People*, *Wreckage*, and *Other*. The final set of categories only applies to images that are part of the *Politics* documents, and the categories are *Meeting*, *Announcement*, *Politician Photographed*, *Civilians*, *Military*, and *Other*. All of these sets of categories are defined in such a way that the categories within a set are mutually exclusive. As already discussed at the start of Section 3.1, I believe that mutually exclusive categories have been under-represented in the text categorization literature, and they tend to be more appropriate when dealing with the news domain.[1]

3.1.2.1 The Manual Categorization Interface

Once a text categorization system is implemented, creating data sets can be the most time-consuming task involved with text categorization research. First, categories must be carefully defined in as unambiguous a manner as possible. Next, documents must be manually labeled with these categories for training and testing.

I have created a user-friendly web interface that allows volunteers to manually label images or entire documents by choosing one label from a set of labels. The interface can easily be updated with new documents or category labels as they are obtained or defined. Volunteers are shown one image and caption (and article, if appropriate) at a time, and they are asked to choose the category that best applies. For each set of categories that I have defined, multiple volunteers have categorized images or documents such that every image or document has been categorized by one volunteer. In addition, I have personally categorized every image

[1]I have also defined an additional set of categories, discussed in Appendix F, that applies only to images and deals with the number of people in each image. This set of categories exists mostly for the researchers at Columbia using image features, and it is rarely used in my own work.

or document. Only if there is agreement between me and the volunteer does the image or document get included in the final data set.[2] For each set of categories, the volunteers are given category definitions and detailed guidelines for choosing a category, although they are also allowed to rely on their intuition. (One of the things I am trying to measure is the level of human agreement.) Appendix B shows a snapshot of the manual categorization tool being used to label a document into one of the *Disaster* image categories (described in Section 3.1.2.4).

3.1.2.2 The Indoor versus Outdoor Data Set

This is the first set of categories that I have explored in detail. There are several reasons for choosing it beyond the general reasons discussed at the start of Section 3.1.2. For one, it is something that can be predicted well both from text and from images features. There is actually a history of image processing research dealing specifically with *Indoor* versus *Outdoor* distinction; researchers in the Electrical Engineering department at Columbia have used the same categories for their image categorization (Paek et al., 1999), and this set of categories has also been used by Szummer and Picard (1998) at MIT for their image analysis. This has lead to the research discussed in Chapter 8, which compares using text against using image features for this task and also examines a combination of the two. Another motivation for examining this set of categories is that there is evidence that humans who cluster images into a hierarchy often use this as a first level of distinction (Vailaya et al., 1999b; Vailaya et al., 1999a). Finally, in my own examination of images, I

[2]At a recent NLP conference, one attendee expressed a criticism of this decision. By only including cases for which there is agreement, I am likely excluding the hardest documents from my data sets. The reason for doing this is that it makes evaluation simpler, since there is a clear right or wrong answer for every prediction. However, there are ways to get around this while still including examples for which volunteers disagree. For example, accuracy rates can be computed separately based on all volunteers and then averaged together; in this case, the best possible accuracy is something under 100%. I still have the labels assigned to images and documents by all volunteers, and I plan to release these with my corpus, so potentially this can still be attempted as future work, either by me or by someone else.

have noticed that the *Indoor* versus *Outdoor* distinction often represents more than just that. For example, in the terrorism domain (in which several researchers from Columbia's NLP research group have worked), *Outdoor* images tend to be at the scene, whereas *Indoor* images tend to be meetings or press conferences.

For the *Indoor* versus *Outdoor* data set, a total of three volunteers have labeled 1,675 images, and I have labeled the same 1,675 images. Five choices exist for each image: *Indoor*, *Likely Indoor*, *Ambiguous*, *Likely Outdoor*, and *Outdoor*. The instructions that have been provided for these categories are presented in Appendix C.1 and can also be found at http://www.cs.columbia.edu/~sable/research/readme.html. 1,339 (79.9%) of the 1,675 images have been given a definite decision in the same direction by both me and the volunteer, and these 1,339 images comprise the primary data set used for the experiments with the *Indoor* and *Outdoor* categories discussed in this thesis. 401 (29.9%) of these images are classified as *Indoor* and 938 (70.1%) are classified as *Outdoor*. A more detailed analysis of the labels selected by volunteers and by me, along with some of the reasons that these categories can be more difficult for humans to choose than one might expect, is provided in Appendix D.

Most of the experiments with the *Indoor* and *Outdoor* categories use the data set consisting of the 1,339 images that have been given a definite agreement by both me and a volunteer. All experiments with this data set use the same breakdown for training and testing. 894 (approximately two thirds) of these 1,339 images have been randomly selected for training, and the remaining 445 images are used for testing. The training set contains 273 (30.5%) *Indoor* images and 621 (69.5%) *Outdoor* images, while the test set contains 128 (28.8%) *Indoor* images and 317 (71.2%) *Outdoor* images. Therefore, a baseline classifier that labels every image as *Outdoor* achieves an overall accuracy of 71.2% (although the F_1 measure of such a classifier would be 0 for the *Indoor* category). For this data set, we also measured human performance when volunteers are shown only the caption of each image and asked to predict the correct category. Humans correctly predict the category for 87.5% of the 1,339 images in the data set, and 87.6% of the 445 images in the test set. We consider this a reasonable upper bound for how well a text categorization system might be expected to perform.

3.1.2.3 The Events Data Set

When I defined the Events data set, the goal was to create categories for news documents with certain properties. First, I wanted the categories to be useful in and of themselves. In other words, I wanted categories that are understandable and intuitive such that it is likely that a user might want to narrow searches or browsings to a specific category. Second, I wanted to define categories that would give us an idea of what types of images to expect in the document. For example, in the next subsection I explain that most images that are part of *Disaster* documents tend to be images of wreckage, victims, rescue workers, etc. Finally, I wanted categories such that each is represented by a reasonable portion of the data, without any one category being too large so as to make it trivially easy to achieve high accuracy. It turns out that the four major categories I have defined (not including the *Other* category) account for close to 95% of the documents.

For the data set involving the Events categories, 28 volunteers have labeled 1,750 documents, and I have labeled the same 1,750 documents. This time, evaluators have been asked to categorize entire documents, each consisting of an image, caption, and article. The choices are the categories themselves: *Struggle*, *Politics*, *Disaster*, *Crime*, and *Other*. The instructions that have been provided for these categories are presented in Appendix C.2 and can also be found at http://www.cs.columbia.edu/~sable/research/instr.html. A total of 1,328 (75.9%) of the 1,750 documents have been assigned identical labels by both me and the volunteer, and these 1,328 documents comprise the data set used for the Events categories. 417 (31.4%) of these documents are classified as *Struggle*, 387 (29.1%) are classified as *Politics*, 296 (22.3%) are classified as *Disaster*, 150 (11.3%) are classified as *Crime*, and 78 (5.9%) are classified as *Other*. See Appendix E for a detailed analysis of labels and human agreement for this category set; briefly, humans show the highest level of agreement by far for the *Disaster* category, which is also the category for which that all systems perform the best, and humans show the lowest level of agreement, by far, for the *Other* category, which is also the category for which all systems perform the worst.

For the experiments with the Events data set, 885 (approximately two thirds) of the 1,328 documents have been randomly selected for training, and the remaining 443 documents are used for testing. The largest category in the training set is *Struggle*, which accounts for 282 (31.9%) of the training documents and 135 (30.5%) of the test documents. Incidentally, the largest category in the test set is *Politics*, which accounts for 243 (27.5%) of the training documents and 144 (32.5%) of the test documents. A baseline classifier that predicts the largest category every time (based on analysis of the training set) would choose *Struggle*, and therefore achieve only a 30.5% overall accuracy.

3.1.2.4 The Disaster Image Data Set

The original reason for defining this set of categories is that I wanted to explore hierarchical categories, since this might have intrinsically led to some novel methods (since there would be the option of categorizing one level at a time instead of directly into nodes). When choosing for which of the Events categories to define subcategories, I eventually selected the *Disaster* category, approximately defined to cover natural disasters and accidents, because existing systems (including my own) achieve almost perfect accuracy (in terms of both precision and recall) for this category (results are given later in this chapter for publicly available systems and in future chapters for my own systems). I have also considered the same properties as when defining the Events categories; i.e. I wanted understandable and intuitive categories that are reasonably represented by the data and that could be used pragmatically to limit searches or browsings. In Chapter 6, I discuss why this data set turns out to be harder than those previously described. Briefly, these categories relate to the focus of an image, requiring that the foreground be distinguished from the background. The mere mention of a concept in a caption (or the mere presence of an object in an image) is not always enough to indicate a category. This creates trouble for any bag of words approach. In Chapter 6, I show that for these reasons, more advanced NLP techniques are therefore necessary to achieve optimal performance.

For the data set involving only the *Disaster* images, a total of four volunteers have labeled the 296 images contained in the documents labeled as *Disaster* (see above description of the Events categories), and I have labeled the same 296 images. The choices given are the categories themselves: *Workers Responding*, *Affected People*, *Wreckage*, and *Other*. The instructions that have been provided for these categories are provided in Appendix C.3 and can also be found at http://www.cs.columbia.edu/~sable/research/instructions.html. 248 (83.8%) of the 296 images have been assigned identical labels by both me and the volunteer, and these 248 documents comprise the data set used for these categories. 98 (39.5%) are classified as *Workers Responding*, 72 (29.0%) are classified as *Affected People*, 55 (22.2%) are classified as *Wreckage*, and 23 (9.3%) are classified as *Other*. 124 (half) of the 248 images have been randomly selected for training, and the remaining 124 images are used for testing. A baseline classifier that chooses the largest category every time would achieve a 39.5% overall accuracy (exactly half of the *Workers Responding* images wound up in the test set).

3.1.2.5 The Politics Image Data Set

This data set was defined with the experiments discussed in Chapter 6 in mind. After discovering that NLP techniques are necessary to achieve reasonable performance for the categorization of *Disaster* images, I decided to create subcategories for one of the other Events categories, and wound up choosing *Politics* because it is easier to identify appropriate categories for these images than for those embedded in *Struggle* or *Crime* documents. As with the *Disaster* Image Data Set, this data set turns out to be hard for systems using standard approaches. The main experiments with these categories are discussed in Section 6.6.

For the data set involving only the *Politics* images, a total of eight volunteers have labeled the 387 images contained in the documents labeled as *Politics* (see above description of the Events categories), and I have labeled the same 387 images. The choices given are the categories themselves: *Meeting*, *Announcement*, *Politician Photographed*, *Civilians*, *Military*, and *Other*. The instructions that have been provided for these categories are provided in Appendix C.4 and can also be found at http://www.cs.columbia.edu/~sable/research/pol_images.html. 299 (77.3%) of the 387 images have been assigned identical labels by both me and the volunteer, and these 299 documents comprise the final data set. 86 (28.8%) are classified as *Meeting*, 64 (21.4%) are classified as *Announcement*, 88 (29.4%) are classified as *Politician Photographed*, 40 (13.4%) are classified as *Civilians*, 14 (4.7%) are classified as *Military*, and only 7 (2.3%) are classified as *Other*. 149 (approximately half) of the 299 images have been randomly selected for training, and the remaining 150 images are used for testing. The largest category in the test set is *Politician Photographed*, which accounts for 30.7% of the test documents. The largest category in the training set, however, is *Meeting*, which only accounts for 27.3% of the test documents, and this would be the performance of a baseline classifier that chooses the largest category every time based on analysis of the training data.

3.1.3 Sample Images

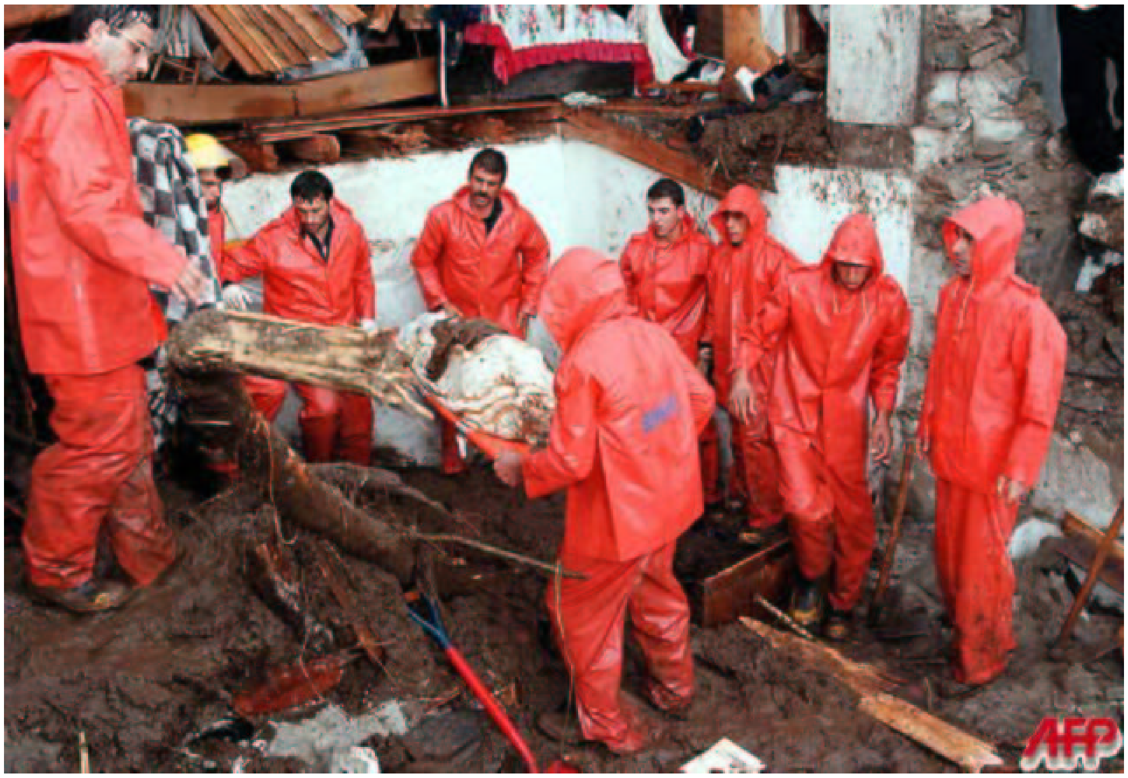
Figure 3.1 and Figure 3.2 show sample images and captions from my corpus. Most captions include a first sentence that describes the associated image and one or two additional sentences that provide background information about the related story. All header information, including locations and dates, is automatically stripped before my experiments. A full sample document, including an article and a captioned image, is shown in Appendix A. Articles generally consist of many paragraphs and are typical in length to what you would expect in a standard newspaper. Based on evidence from the experiments discussed in Appendix G, only first sentences of captions are used for the *Indoor* versus *Outdoor* experiments, and also for experiments involving the *Disaster* image data set and the *Politics* image data set, but entire articles are used for experiments involving the Events data set.

3.1.4 How to Obtain the Corpus

The corpus just described is now ready to be made public. Once this happens, I will post instructions at http://www.cs.columbia.edu/~sable/research/corpus.html. I believe that this corpus may serve as an important resource for future researchers. To the best of my knowledge, it will be the only public text categorization corpus containing images, and as I will demonstrate throughout the rest of this thesis, this has led to some very interesting results.

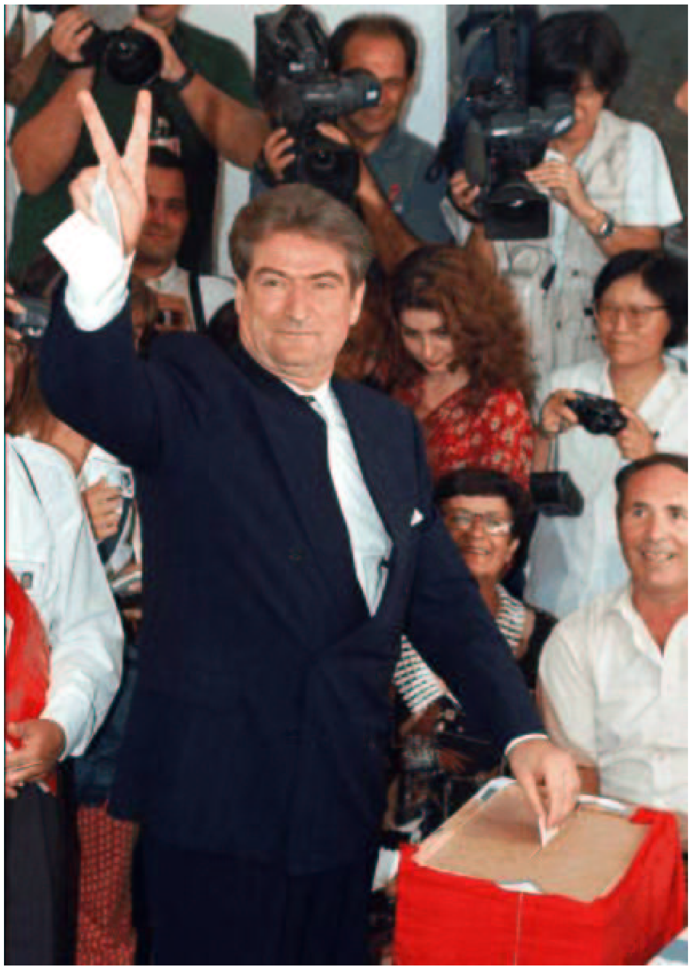
3.2 Results and Evaluation of Pre-existing Systems

The publicly available Rainbow package (McCallum, 1996), described in Section 2.8, actually consists of several text categorization systems relying on a variety of ap-



PONTA DELGADA, PORTUGAL, 1-NOV-1997: Rescue workers remove the body of a man from mud, October 31, following a landslide in Ribeira Quente, Azores. Ten people died and dozens are still missing inside their destroyed houses after a mass of rocks fell on a group of homes. The disaster may have been caused by heavy rain which has battered the Azores. [Photo by AFP]

Figure 3.1: The categories associated with this image and its document are *Outdoor*, *Disaster*, and *Workers Responding*.



TIRANA, ALBANIA, 29-JUN-97: Albanian President Sali Berisha casts his vote at a central Tirana polling station on, June 29. Albania holds its general elections three months after the collapsed pyramid investment schemes drove the country into armed turmoil. [Photo by Petr Josek, Reuters]

Figure 3.2: The categories associated with this image and its document are *Indoor*, *Politics*, and *Politician Photographed*.

proaches. The approaches used by the Rainbow systems are Rocchio/TF*IDF, K-Nearest Neighbor, Naive Bayes, Probabilistic Indexing, Maximum Entropy, and Support Vector Machines; these approaches have all been explained in Sections 2.6.1 and 2.6.2. All of the Rainbow systems are implemented in such away that they assume mutually exclusive categories; they are, therefore, appropriate for the data sets I have created.

In this section, I describe the performance of the Rainbow systems for the data sets that comprise my corpus, and in later chapters, I compare these results to the results of systems I have created. It should be noted that I have used the Rainbow systems with their default settings. Although the Rainbow systems are being trained using the training sets of my own data sets, there are certain global parameters that have been decided without taking into account this specific corpus (e.g. tokenization rules, referring to exactly what constitutes a word, which can vary as discussed in Section 2.4.1, or for SVMs, the kernel function, as described in Section 2.6.2.3). Since I have developed my own systems with my own corpus in mind, it is possible that the Rainbow systems are at some sort of disadvantage, and this should be kept in mind when evaluating my own systems.

3.2.1 Results for the Indoor versus Outdoor Data Set

Table 3.1 shows the results of the Rainbow systems when tested on the *Indoor* versus *Outdoor* data set. For this data set, only first sentences of captions are used, as explained in Section 3.1.3. The first column, representing overall accuracy, is the most important. The next two columns show the F_1 results (as defined in Section 2.7) for each category. As can be seen, all systems achieve reasonably respectable performance, although the K-Nearest Neighbor system is well behind the others. The best system is the Probabilistic Indexing system, followed by the Naive Bayes system. According to a one-sided χ^2 test, the performance of each

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
Rainbow systems			
Naive Bayes	85.4	73.5	89.9
Rocchio/TF*IDF	84.5	73.2	89.1
K-Nearest Neighbor	77.8	65.3	83.6
Probabilistic Indexing	86.3	78.1	90.0
Support Vector Machines	82.0	66.9	87.7
Maximum Entropy	84.5	70.9	89.4

Table 3.1: Rainbow systems perform well for the *Indoor* versus *Outdoor* data set.

system except the bottom two (SVMs and kNN) falls within the 95% confidence interval of the performance of the best system (Probabilistic Indexing).[3] In this case, the systems relying on newer, more advanced techniques (SVMs and Maximum Entropy) do not perform quite as well as the top two systems. Remember that a baseline system that chooses the larger category every time would achieve a 71.2% overall accuracy, while the upper bound set by humans who have predicted each image's category based only on its caption is 87.6% (as discussed in Section 3.1.2.2). That does leave some room for improvement, even for the best system, and this is explored in detail in Chapter 8.

3.2.2 Results for the Events Data Set

Table 3.2 shows the results of the Rainbow systems when tested on our Events data set. For this data set, full articles are used, as explained in Section 3.1.3. As is the case for the *Indoor* versus *Outdoor* data set, all systems achieve respectable performance, although the K-Nearest Neighbor system is once again well behind the rest. This time, the two best systems are the two most advanced (SVMs and Maximum Entropy). The Naive Bayes system comes in a respectable third

[3]To perform significance tests for this thesis, I have used the prop.test function in S-PLUS.

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
Rainbow systems						
Naive Bayes	87.6	86.2	86.3	96.7	89.1	61.5
Rocchio/TF*IDF	87.4	81.1	85.3	97.7	88.4	68.3
K-Nearest Neighbor	81.9	80.0	79.7	95.6	75.6	63.2
Probabilistic Indexing	86.5	83.6	84.8	97.2	89.4	65.0
Support Vector Machines	88.7	88.1	89.2	96.2	87.0	57.9
Maximum Entropy	88.3	88.1	87.9	95.7	87.9	55.6

Table 3.2: Rainbow systems perform well for the Events data set.

place, once again appearing near the top. According to a one-sided χ^2 test, the performance of each system except the bottom one (kNN) falls within the 95% confidence interval of the performance of the best system (SVMs). All systems perform way above the baseline of 30.5%. Note further that all systems do at least somewhat well for all categories, with all systems doing the best for the *Disaster* category and the worst for the *Other* category, while the order of performance for the other three categories varies from system to system. In Appendix E, I show that humans, interestingly enough, have the highest rate of agreement, by far, for the *Disaster* category and the lowest rate of agreement, by far, for the *Other* category, whereas the rates of agreement for the other three categories are very similar to each other.

3.2.3 Results for the Disaster Image Data Set

Table 3.3 shows the results of the Rainbow systems when tested on the *Disaster* image data set. For this data set, only first sentences of captions are used, since they describe the image, and the words in other sentences are almost certainly not important (see Chapter 6 for a detailed support of this argument for these categories). This time, all of the systems perform poorly, although still above

System	Overall Accuracy %	Workers Responding F_1 %	Affected People F_1 %	Wreckage F_1 %	Other F_1 %
Rainbow systems					
Naive Bayes	55.6	66.7	54.3	45.9	16.7
Rocchio/TF*IDF	54.0	66.0	50.0	50.0	22.2
K-Nearest Neighbor	54.0	63.2	55.6	43.1	0.0
Probabilistic Indexing	59.7	68.1	55.2	61.3	28.6
Support Vector Machines	54.8	64.6	51.4	50.0	33.3
Maximum Entropy	58.1	69.2	64.8	43.3	0.0

Table 3.3: Rainbow systems perform poorly for the *Disaster* image data set.

the baseline of 39.5%, with performance ranging from 54.0% (Rocchio/TF*IDF and kNN) to 59.7% (Probabilistic Indexing). According to a one-sided χ^2 test, the performance of every system falls within the 95% confidence interval of the performance of the best system (Probabilistic Indexing). All systems do the best (although still not great) for the *Workers Responding* category, and the worst for the *Other* category (two systems actually have a 0 F_1 measure for this category). Here we see the first example of a set of categories that seems to defy standard approaches to text categorization. The reasons are not obvious; I show in Chapter 6 that humans who view only the first sentences of captions are able to predict the correct category over 90% of the time. I also show in that chapter, however, that most of the words are not useful, and that syntax - which is ignored by all standard systems - plays a key role.

3.2.4 Results for the Politics Image Data Set

Table 3.4 shows the results of the Rainbow systems when tested on our *Politics* image data set. As with the *Disaster* image data set, only first sentences of captions are used, for similar reasons. Once again, all of the systems perform poorly, although this time there is more of a range. At the bottom is the K-Nearest Neigh-

System	Overall Accuracy %	Meeting F_1 %	Announcement F_1 %	Politician Photographed F_1 %	Civilians F_1 %	Military F_1 %	Other F_1 %
Rainbow systems							
Naive Bayes	53.3	64.9	49.2	51.2	41.4	0.0	0.0
Rocchio/TF*IDF	54.7	71.9	52.6	53.0	36.4	20.0	22.2
K-Nearest Neighbor	36.0	50.0	25.0	17.2	20.0	0.0	0.0
Probabilistic Indexing	64.0	77.6	73.0	58.7	44.4	30.8	15.4
Support Vector Machines	56.7	72.0	52.2	55.6	40.0	0.0	0.0
Maximum Entropy	53.3	59.8	47.9	58.1	48.3	0.0	0.0

Table 3.4: Rainbow systems perform poorly for the *Politics* image data set.

bor system, with a mere 36.0% overall accuracy, not far above the baseline of 27.3%. At the top is the Probabilistic Indexing system, which achieves an overall accuracy of 64.0%, bordering on respectable. According to a one-sided χ^2 test, the performance of the Probabilistic Indexing system is significantly better than that of the other Rainbow systems (the largest p-value is 4.3%). Excluding the best and worst system, however, you can see that the remaining systems all show very similar performance for this data set. All systems do the best for the *Meeting* category (three systems achieve an F_1 measure above 70% for this category, which is somewhat respectable), while two categories present major difficulties, those being *Military* and, once again, the elusive *Other*. This set of categories is discussed further in Section 6.6.

3.3 Research Related to the Categorization of Images Using Text

There is very little in the pre-existing literature that directly and explicitly discusses the categorization of images using text. Two exceptions are (Smith and Chang, 1996a) and (Smith and Chang, 1997). These articles are mainly about automatic search for images and video using both text and image features. In addition, there is a section on the automatic classification of images into a taxonomy using keywords extracted from URLs, hyperlinks, and "alt" tags. These extracted keywords are compared to entries in a semi-automatically created dictionary that maps terms to categories.

Other researchers attempt to categorize images into certain sets of categories based solely on image features. Szummer and Picard (1998) attempt to categorize images as *Indoor* or *Outdoor* using low-level features such as color histograms. The research discussed in (Paek et al., 1999) concerns a colleague's efforts involving the use of similar low-level image features for the same task, my own work using text for this task, and also an attempt to integrate the two two approaches together. I focus more on the *Indoor* versus *Outdoor* categorization task in Chapter 8, in which I discuss some of my more current research comparing the use of text against the use of image features, and also examine a combination of both, for this task.

The work discussed in (Vailaya et al., 1999b) and (Vailaya et al., 1999a) involves the use of similar low-level image features to divide *Outdoor* vacation images into *City* images and *Landscape* images, and they further consider dividing the *Landscape* images into the categories *Sunset*, *Forest*, or *Mountain*. At all levels (*Indoor* versus *Outdoor*, *City* versus *Landscape*, and *Sunset* versus *Forest* versus *Mountain*), the authors recognize the existence of an *Other* category, which also occurs throughout this thesis (although I do not consider the same sets of categories as they do). In their work, all sets of categories are motivated by experiments with human subjects who evaluated 171 vacation images and were asked to group the images into meaningful categories.

Many systems have been developed for image retrieval based on image features. Just a few of these that I have come across in my research include QBIC (Niblack et al., 1993), Photobook (Pentland, Picard, and Sclaroff, 1994), Foureyes (Picard and Minka, 1995), VisualSeek (Smith and Chang, 1996c), WebSEEk (Smith and Chang, 1997), and MARS (Hehrotra et al., 1997). Of these, only WebSEEk also uses text. Benitez and Chang (2002b; 2002a) discuss the use of both text and low-level image features to cluster images. The potential for using these clusters to aid in categorization is discussed but not implemented.

Several researchers have used text from image captions to aid in image retrieval (Smeaton and Quigley, 1996; Rowe and Guglielmo, 1996; Elworthy, 2000). In all of these works, some NLP techniques are used to compare a user's query to captions from an image database. Srihari (1995) has developed a system called Piction, which matches names in image captions to faces in the image, also for the purpose of retrieval. The work discussed in (Duygulu et al., 2002) involves the use of the expectation maximization (EM) algorithm and a process analogous to learning a lexicon from aligned bilingual text in order to automatically annotate regions of images with provided keywords.

3.4 Concluding Discussion of Categorizing Images Using Text

I have shown in this chapter that pre-existing text categorization systems can be applied to text associated with images to categorize the images. In order to demonstrate this ability, I have created a new corpus consisting of news documents with embedded captioned images. I have carefully defined multiple sets of categories representing various levels of abstraction, and I have collected manual labels from volunteers (as well as myself) using a user-friendly, web-based interface that I have implemented. This corpus, to the best of my knowledge, is the only public text categorization corpus containing images.

I have tested six systems, using a variety of common text categorization methodologies, that comprise the publicly available Rainbow package. In terms of comparing the various systems to each other, no definite conclusions can be made, although some systems clearly do better than others. The Probabilistic Indexing has the best performance for three of the four data sets (the exception being the Events data set, for which it has the second worst performance). The K-Nearest Neighbor system has the worst (or tied for the worst) performance for every data set. However, we should be not too hasty to dismiss this approach; in detailed studies by Yang (1999) or Yang and Liu (1999), a kNN system performs near the top of a pack of many competing systems for Reuters data sets. In the study by Yang and Liu, a Support Vector Machine system performs best. In my experiments, the SVM system included with Rainbow does reasonably well, exhibiting the best performance for the Events data set, but placing in the middle of the pack for the rest. It's worst performance is for the *Indoor* versus *Outdoor* data set. Although I have not previously shown the results in this chapter, I have also tested SVM^{light}, the same SVM system used by Yang and Liu in their experiments, for the *Indoor* versus *Outdoor* data set. The overall accuracy is 85.6%, which is significantly better than Rainbow's SVM system, but still not as good as Rainbow's Probabilistic Indexing system or some others of my own creation that are described in later chapters. The Rainbow system using the advanced Maximum Entropy methodology also has reasonable performance, coming in second for two sets categories and somewhere in the middle for the rest. Even the Rocchio/TF*IDF system, though, does reasonably well, showing up in the middle of the pack for these data sets. Recently, most works comparing methods, such as the one just cited by Yang, place this method near the bottom, and it is mostly used as a baseline these days. I show, however, in Chapter 4, that the application of a statistical technique known as density estimation to the results of my own Rocchio/TF*IDF system makes it very competitive. Finally, the system using a Naive Bayes methodology, which is also often used as a baseline, performs reasonably well, falling in the middle of the pack for every data set. In Chapter 5, I show that a generalization of Naive Bayes relying on the concept of binning as a smoothing technique is extremely competitive, beating all other systems that I have tested for my first two data sets (*Indoor* versus *Outdoor* and Events).

For some sets of categories, such as *Indoor* versus *Outdoor* or the Events categories, the existing, standard systems perform well without modification. In Chapter 8, I discuss attempts to improve performance even more for the *Indoor* versus *Outdoor* data set by relying on a combination of text and image features. (Image features are obviously not typically available for most text categorization tasks discussed in the literature, and even for my other data sets, the state-of-the-art is not good enough to perform well for these categories.) The other sets of categories I have defined, those that apply to the Disaster images and Politics images, defy standard text categorization systems. This is discussed in great detail in Chapter 6, where I explain why these categories are hard, and what can be done about it. Briefly, though, the characteristics of the text and categories involved are quite different than those involved with most text categorization tasks. In this case, syntax becomes very important, and standard text categorization techniques ignore that. So, while I have shown in this chapter that text categorization techniques are useful for labeling images, for certain tasks involving specific sets of categories, more work is necessary to achieve optimal performance.

Chapter 4

Density Estimation

As is evident from Chapter 2, there are a myriad of methods used for text categorization, and although some researchers may claim differently, there is no clear "winner" in all cases. When I first started my text categorization research, I was not content to simply use existing methodologies, but was searching for ways to improve them. The first thing I tried was using cross validation experiments to test various features in order to discover which tend to be important, at least for a particular task. This work, discussed in Appendix G, eventually led to the research discussed in this chapter, which involves the use of a statistical technique known as density estimation (Silverman, 1986). Density estimation is used in conjunction with other systems that rely on existing methodologies to produce better results.

In this chapter, I present my research on the use of density estimation to potentially improve the results of certain text categorization methods. The methods that are eligible are those that label documents by computing a similarity measure (or other score, such as a probability estimate) for every document/category pair. These methods include Rocchio/TF*IDF, Naive Bayes, K-Nearest Neighbor, certain implementations of Support Vector Machines, and many other commonly used techniques. Density estimation converts vectors of such similarity measures to probabilities of category membership. These probabilities provide confidence measures of systems' predictions, and I show, through experiments, that the technique also often improves the accuracy of a system.

The system to which I have applied density estimation is that described in (Sable and Hatzivassiloglou, 2000) and (Sable and Hatzivassiloglou, 1999). This system, which I have developed, applies a Rocchio-based method involving TF*IDF text similarity measures as described Section 2.6.1.1 in conjunction with novel features, such as consideration of part-of-speech and different spans of text. An in-depth discussion of these novel features can be found in Appendix G. Standard Rocchio/TF*IDF computes a similarity measure for every document/category pair and makes its decisions based on these measures (e.g. by choosing the category with the highest score). As discussed in Section 2.6.1.1, the Rocchio approach is often used as a baseline for text categorization (Joachims, 1997; Lewis et al., 1996; Schapire and Singer, 2000; Yang, 1999), although in recent years, advanced methods such as Support Vector Machines (SVMs) (Joachims, 1998) have achieved significantly better results.

This chapter explains how density estimation can be applied to the results of systems such as the one I have implemented to convert numerical similarity scores for categories to probabilities of membership in each category by estimating the proportion of documents with similar scores in the training set that fall into the category. A Rocchio-based system is an ideal type of system to which density estimation can be applied because it computes similarity scores for categories that do not already have any intrinsic meaning. I show that density estimation, in addition to providing confidence measures for predictions, improves the accuracy of my system in the majority of cases, boosting results to surpass those of several advanced methods. Density estimation is a technique that can be applied to the results of any system that computes, for every test document, a score for every category.

4.1 Description of Density Estimation

Density estimation is a statistical technique for estimating a probability density[1] for the distribution assumed to generate a set of empirically obtained data points. My approach to using density estimation for the purposes of text categorization can be applied in conjunction with any text categorization system that expresses a similarity score between each document and category. It works regardless of the number of categories, and it works with either mutually exclusive categories or with independent, binary categories.

Assume we are dealing with a set of N categories (c_1, c_2, ..., c_N). Let us also assume that we have a text categorization system using a method that can assign to any given document d a similarity score to category c_i, namely $S_{(d,c_i)}$. This can be done for every category, and so we can obtain for each document a vector of similarity scores, one for each category. The vector for some given document d can be represented as $V_d = [S_{(d,c_1)}, S_{(d,c_2)}, ..., S_{(d,c_N)}]$.

Rocchio/TF*IDF systems create just such a set of similarity scores for every document. These scores only have meaning in comparison with each other, and so this constitutes an ideal type of system to which density estimation can be applied. Systems that use methods such as Naive Bayes or K-Nearest Neighbor generally compute probabilities of category membership for each category. These probabilities can be used in place of similarity scores in the vector above, and density estimation can be used to re-scale the probabilities. SVMs, in their basic form, make binary (YES/NO) decisions for individual categories by deciding whether a transformation of a representation of a document falls on a particular side of a hyperplane in a mapped vector space. When used for mutually exclusive categories,

[1]Or a probability mass function, if the distribution is assumed to be discrete.

the distances from the hyperplanes are sometimes used to determine positive or negative (depending on the side of the hyperplane) scores for categories, and these scores could be used for the vectors to which density estimation could apply. Density estimation potentially can be applied to most commonly used techniques for text categorization; exceptions include decision tree approaches and expert systems that follow a specific chain of rules that ends in a prediction of a specific category without generating scores for each category.

When dealing with mutually exclusive categories, a standard text categorization system, after generating a vector of category similarity scores for a document, would assign the document to the category with the highest score. While this seems intuitive, it is not always the best solution, because the similarity measures do not always have an intrinsic meaning, and the scale is not always the same for every category. When dealing with binary categories, there are several standard methods of converting similarity scores to YES/NO decisions for every possible document/category pair. One, known as Scut (Yang, 1999), involves determining the optimal threshold for each category based on training data. (Optimality can be determined, for example, by maximizing F_1 measures, as described in Section 2.7). Another method, known as Pcut (Yang, 1999), involves measuring the percentage of training documents that falls into each category in the training set and assuming that a similar percentage falls into the category in the test set. Therefore, for each category, we assign the category to the x test documents with the highest similarity, where x is chosen based on the training set. A third method is to create, for each category, a separate category consisting of all documents not in the actual category. To obtain a YES/NO decision for the actual category, we compare the similarity score of a test document and the actual category to the similarity score of the test document and the created category, thus converting our multi-label, binary categorization task to a set of categorization tasks each with two mutually exclusive categories.

The use of density estimation replaces those methods described in the previous paragraph. First, the system is used not only to obtain similarity score vectors for each test document, but also for each training document. (Of course, the vectors for the training documents only need to be calculated once ahead of time.) So, a vector of category similarity scores, as described above, is computed for every training document as well as every test document. Let d_1 and d_2 be two documents in the corpus, and $S_{(d,c)}$ be the similarity score assigned to a document d for a category c. We can then measure the Euclidean distance between the similarity score vectors for d_1 and d_2 as follows:

$$D_{(d_1,d_2)} = \sqrt{\sum_{i=1}^{N}[S_{(d_1,c_i)} - S_{(d_2,c_i)}]^2} \tag{4.1}$$

To use density estimation to label a test document d, distances are computed between the document's similarity score vector and those of every training document. The k training documents with the closest similarity score vectors to that of d, and thus the smallest distances according to the above formula, are selected. In other words, we are choosing training documents whose similarity score vectors fall within an N-dimensional hypersphere centered at the point specified by the similarity score vector for d. The labels of these document are used to determine the predicted label of the test document as described below. My implementation of density estimation selects the same number of training documents for each test document (as opposed to examining a hypersphere of fixed radius) due to the potential variability in sparseness of data for different values of similarity scores. The parameter k (the number of training documents used) can be chosen arbitrarily, or it can be determined based on cross validation experiments within the training set.

Once training documents with similarity score vectors close to that of d have been identified, density estimation provides a probability estimate of membership in each category c, estimated as the proportion of those neighbors from the training set that are assigned the label c. In calculating this proportion, individual training documents are weighted inversely proportional to the distance of their similarity score vectors from the center of the hypersphere (corresponding to the similarity score vector of the test document d). A separate decision can be made for each category based on the (weighted) percentage of close training documents that have been assigned to the category, and probability estimates for all possible categories can be assigned to the test document. More formally, let d_i be the i^{th} training document out of the k training documents selected as described above, and let $I_{(d_i,c)}$ be 1 if d_i belongs to category c and 0 otherwise. The estimated probability that the current document d belongs to some specific category c is:

$$P(c|d) = \frac{\sum_{i=1}^{k} I_{(d_i,c)} \frac{1}{D_{(d,d_i)}+\epsilon}}{\sum_{i=1}^{k} \frac{1}{D_{(d,d_i)}+\epsilon}} \tag{4.2}$$

Note that the numerator and denominator in the above formula are the same except for the $I_{(d_i,c)}$ term, so that documents that belong to category c contribute to both the numerator and denominator, and documents that do not belong to category c contribute only to the denominator. The epsilon in the formula is just an arbitrary, very small constant to avoid infinities in the case that there is some training document with a similarity score vector that exactly matches that of d.

Density estimation as just described can be applied with any number of categories and it can be used as described regardless of whether we are dealing with mutually exclusive categories or binary categories. If we are dealing with mutually exclusive categories, every selected training document has exactly one label, and the probabilities assigned to categories for a given test document add up to exactly 1. If we are dealing with binary categorization, each individual category is assigned a probability ranging from 0 to 1, but there is no further restriction on the sum of these probabilities (since the categories are independent, and a document may belong to more than one category or none at all). When dealing with binary categorization, density estimation is applied to the raw similarity scores between a test document and categories; this method replaces the use of Scut, Pcut, or conversion to a set of mutually exclusive categorization tasks described earlier in the section.

You might have noticed a strong resemblance between K-Nearest Neighbor approaches, as described in Section 2.6.1.2, and density estimation. In each case, we are choosing certain training examples and using their categories to predict the category or categories of a test document. The important distinction is that density estimation is *not* comparing the actual test document to any training documents. It is only comparing the category similarity score vector of the test document to category similarity score vectors of training documents. It is very possible that a document that might not share any words or terms in common with a test document may have a very similar (maybe even an identical) category similarity score vector. The purpose of density estimation is not to find training documents that are similar to the test document, but rather to interpret the category similarity scores of the test document which can be the result of some entirely different system. For each experiment discussed in this chapter, I do compare the results of my system using density estimation to those of one or more kNN systems.

4.2 Experiments with Density Estimation

I have evaluated the benefits of density estimation through experimentation on three data sets taken from two corpora. The first data set is the *Indoor* versus *Outdoor* data set which I describe in Section 3.1.2.2. The second data set is the Events data set which I describe in Section 3.1.2.3. The third data set is the ModApte split of the Reuters-21578 (Lewis, 1997) corpus, which has already been described in Section 2.8. Whereas the first two data sets come from the corpus that

I have created, and I have defined the categories involved to be mutually exclusive, the third task data set involves binary categories (i.e. each document can have any number of labels up to the number of categories).

For each of the data sets just mentioned, I first apply my Rocchio/TF*IDF system and then I apply density estimation to the results of the system to measure its effect on performance. My system allows many parameters that are not typical of other Rocchio-based systems (e.g. the restriction to words of specific grammatical categories as determined by a statistical part-of-speech tagger (Church, 1988)); these parameters are discussed in detail in Appendix G. My system automatically performs cross validation experiments within the training set to choose the settings for the optional parameters that are likely to maximize performance. For the experiments discussed in this chapter, I have performed these cross validation experiments with and without density estimation, as it is possible that the use of density estimation may change the optimal settings for certain other parameters. For example, one of the parameters deals with normalization of category word vectors, which becomes more important when density estimation is not used since both normalization and density estimation can account for skewed category sizes. For the cross validation experiments using density estimation, I also try out multiple possible values of k, which, as described in Section 4.1, is the number of training documents used to predicted the label or labels of each test document.

In addition to measuring the effect of density estimation on the performance of my system for each data set, I also compare the results to those of the competing systems. For the first two experiments, I therefore compare against the Rainbow systems, for which I have already reported results in Section 3.2. Since I was particularly interested in comparing results with those of a K-Nearest Neighbor system, and that which is part of the Rainbow package seems to perform poorly, I have also implemented my own version of a kNN system that uses the same word vectors for documents as does my Rocchio/TF*IDF system. For the third experiment, which uses binary Reuters categories, I can not test the Rainbow systems or my own version of a kNN system because these have been implemented to handle only data sets with mutually exclusive categories. Instead, I compare results against all systems tested by Yang and Liu (1999) (including a kNN system) in a controlled study conducted using the same data set. These systems have been chosen because they use well-known methods and achieve strong performance scores in previously reported studies discussing similar experiments (Yang and Liu, 1999).

For the first two experiments, the main metric I consider is overall accuracy. Each test document is assigned one category, and the overall accuracy of the system is the percentage of such assignments that are correct. I also report the F_1 measures, as described in Section 2.7, for each category. For the third experiment, to allow for direct comparison with Yang and Liu (1999), I report micro-averaged precision, micro-averaged recall, micro-averaged F_1, macro-averaged F_1, and overall error (which is 1 minus overall accuracy). All of these metrics have also been defined in Section 2.7. I focus on the micro-averaged F_1 score which has been more widely used for comparing methods and systems than the macro-averaged alternative (Yang and Liu, 1999); as discussed in Section 2.7, this measure makes more sense than overall accuracy for tasks involving binary categories.

4.3 Results and Evaluation of Density Estimation Experiments

4.3.1 Results for the Indoor versus Outdoor Data Set

Using the chosen parameters based on the cross validation experiments discussed in Section 4.2 (with and without density estimation), I have trained my system on

	Actual Indoor	Actual Outdoor	Precision
System Indoor	100	61	62.1%
System Outdoor	25	259	91.2%
Recall	80.0%	80.9%	

Table 4.1: Results for the *Indoor* versus *Outdoor* data set before density estimation.

	Actual Indoor	Actual Outdoor	Precision
System Indoor	87	24	78.4%
System Outdoor	38	296	88.6%
Recall	69.6%	92.5%	

Table 4.2: Results for the *Indoor* versus *Outdoor* data set after density estimation.

the entire training set and tested on the previously unseen test set. The overall accuracy of my system when density estimation is *not* used is 80.7%. Table 4.1 shows the precision and recall values achieved on the test set for each category. The F_1 measures for the *Indoor* and *Outdoor* categories are 69.9% and 85.7% respectively. The overall accuracy of my system after density estimation is applied rises to 86.1%.[2] Table 4.2 shows the precision and recall values achieved on the test set for each category. The F_1 measures for the *Indoor* and *Outdoor* categories are 73.7% and 90.5% respectively. So density estimation not only has a statistically very significant effect on the overall accuracy of the system (a one-sided χ^2 test indicates a p-value of 0.25%), but it also improves performance (based on the F_1

[2]Whichever category has the highest probability according to density estimation is considered to be the system's prediction.

measure) for both categories.

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
My systems			
Rocchio/TF*IDF	80.7	69.9	85.7
Density Estimation	86.1	73.7	90.5
K-Nearest Neighbor	82.7	65.8	88.4
Rainbow systems			
Naive Bayes	85.4	73.5	89.9
Rocchio/TF*IDF	84.5	73.2	89.1
K-Nearest Neighbor	77.8	65.3	83.6
Probabilistic Indexing	86.3	78.1	90.0
Support Vector Machines	82.0	66.9	87.7
Maximum Entropy	84.5	70.9	89.4

Table 4.3: Density estimation leads to a significant increase in accuracy for the *Indoor* versus *Outdoor* data set.

Table 4.3 shows the results of all systems tested on the first data set. As mentioned in the previous paragraph, density estimation leads to a significant increase in accuracy for this experiment. In addition, my system with density estimation performs better than all but one competing system, that being a Probabilistic Indexing system that beats my system with density estimation by only one test document. Of the seven systems that do not perform as well, the performance of three falls within a 95% confidence interval. All systems beat a baseline of 71.2% accuracy, which could be achieved by a system that chooses the largest category every time. For this data set, I have also measured the performance of humans who are asked to predict whether the images in the test set are *Indoor* or *Outdoor* by looking only at the textual captions. The overall accuracy of humans is 87.6%, and I consider this to be a reasonable upper bound for how well an automatic system using only text might be expected to do. Note that the best systems, including my system with density estimation, are within 2% of this result.

4.3.2 Results for the Events Data Set

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
My systems						
Rocchio/TF*IDF	87.1	85.0	88.4	98.8	79.2	60.0
Density Estimation	84.9	83.7	86.0	97.3	80.0	34.3
K-Nearest Neighbor	84.0	81.1	82.1	93.9	81.3	65.0
Rainbow systems						
Naive Bayes	87.6	86.2	86.3	96.7	89.1	61.5
Rocchio/TF*IDF	87.4	81.1	85.3	97.7	88.4	68.3
K-Nearest Neighbor	81.9	80.0	79.7	95.6	75.6	63.2
Probabilistic Indexing	86.5	83.6	84.8	97.2	89.4	65.0
Support Vector Machines	88.7	88.1	89.2	96.2	87.0	57.9
Maximum Entropy	88.3	88.1	87.9	95.7	87.9	55.6

Table 4.4: Density estimation degrades performance for the Events data set.

On this particular data set, my Rocchio/TF*IDF system achieves better performance without density estimation. Without density estimation, the overall accuracy of the system is 87.1%. The F_1 measures for the categories *Struggle*, *Politics*, *Disaster*, *Crime*, and *Other* are 85.0%, 88.4%, 98.8%, 79.2%, and 60.0% respectively. With density estimation, the overall accuracy is 84.9%. The F_1 measures for the categories *Struggle*, *Politics*, *Disaster*, *Crime*, and *Other* are 83.7%, 86.0%, 97.3%, 80.0%, and 34.3% respectively. Therefore, density estimation degrades the overall accuracy for this data set by 2.2%, and performance is worse for four of the five Events categories according to F_1 measures.

This does not necessarily mean that a user would not want to use density estimation. The difference in overall accuracy is not statistically significant (a one-sided χ^2 test indicates a p-value of 10.6%), and using density estimation assigns confidence measures in terms of probability to all predictions, whereas with a standard approach, the best one can get is a ranked list of categories. Appendix H provides an analysis of the validity of the confidence measures assigned by density estimation for the experiments involving the *Indoor* versus *Outdoor* data set and the Events data set. Also, as discussed in Section G.3, it appears that density estimation would improve the accuracy of this system when applied to the Events data set if the other features of the system were not set just right.

For further comparison, I have tested all of the same systems used in the previous experiment. Table 4.4 shows the results. In this experiment, several of the competing systems have a higher overall accuracy than my own. All competing systems except the top two (SVMs and Maximum Entropy) and the bottom one (Rainbow's kNN) achieve a performance that falls within the 95% confidence interval of the performance of the density estimation system. All systems far outperform the baseline of 30.5%, and my system with density estimation still beats both versions of kNN systems tested.

4.3.3 Results for the Reuters Data Set

Method	miR %	miP %	miF1 %	maF1 %	error %
My systems					
Rocchio/TF*IDF (Pcut)	71.21	70.72	70.96	50.14	0.803
Density Estimation	78.93	87.48	82.98	40.52	0.446
Combo	80.48	83.90	82.15	51.18	0.482
Tested by Yang and Liu					
Support Vector Machines	81.20	91.37	85.99	52.51	0.365
K-Nearest Neighbor	83.39	88.07	85.67	52.42	0.385
Linear Least-Squares Fit	85.07	84.89	84.98	50.08	0.414
Neural Network	78.42	87.85	82.87	37.65	0.447
Naive Bayes	76.88	82.45	79.56	38.86	0.544

Table 4.5: Density estimation leads to a tremendous improvement for the experiment involving a Reuters data set.

Table 4.5 shows the results of all systems tested on the Reuters data set. The five columns of results represent micro-averaged recall, micro-averaged precision, micro-averaged F_1, macro-averaged F_1, and overall error (equal to 1 minus overall accuracy). The bottom five rows of the table are a reproduction of Table 1 from (Yang and Liu, 1999), summarizing the results of all categorizers that Yang and Liu tested. The five methods used are Support Vector Machines, K-Nearest Neighbor, Linear Least-Squares Fit, Neural Network, and Naive Bayes. They are ranked from top to bottom in the table based on micro-averaged F_1, generally considered to be the most important of these performance measures (Yang and Liu, 1999).

The top row of the table shows the results of my system without density estimation. Without density estimation, some other technique is needed to convert category similarity scores to YES/NO decisions for each category as described in Section 4.1, since Reuters is a binary categorization corpus. I have tried all three of the standard methods described in Section 4.1, and Pcut works by far the best, so that is used for the result shown in the first line of the table. The micro-averaged F_1 measure, the one that I try to optimize, is only 70.96%, far lower than the five methods tested by Yang and Liu. This is not surprising, as standard Rocchio does not generally fare well against other methods for binary categorization. For example, see Yang (1999) in which she includes Rocchio among the compared systems tested on similar Reuters data sets. Interestingly, the macro-averaged F_1 measure is higher than three of the five systems tested by Yang and Liu. (This demonstrates how results can be misleading if researchers choose an evaluation metric after obtaining results.)

The second row of the table shows the results of my system after density estimation is applied. As can be seen, the micro-averaged F_1 goes up from 70.96% to 82.98%, a major improvement. In addition, the overall error goes down from 0.803% to 0.446%. These scores are better than those for the Naive Bayes approach and marginally better than those for the Neural Network approach examined by Yang and Liu. Density estimation takes a system that is far under-performing top competing systems and improves its performance to such a degree that it is in the pack.

Interestingly, density estimation actually lowers the macro-averaged F_1 of my system from 50.14% to 40.52%. This leads me to believe that density estimation may perform better for large categories while the standard approach may perform better for small categories (since macro-averaged F_1 treats all categories equally). I have therefore performed a final experiment combining the two approaches. For all categories for which density estimation performs better according to F_1 measures based on cross validation experiments within the training set, I also use density estimation for the test set, but for other categories, I use Pcut for the test set. It turns out that Pcut is used for 52 of the 90 categories, but these categories account for only 1,257 assignments in the training set, whereas density estimation is used for only 38 categories, but these categories account for 8,626 assignments in the training set. The results of this final experiment are summarized in row 3 of Table 4.5. As can be seen, the micro-averaged F_1 and overall error are slightly worse than when density estimation is used for all categories, but still much better than Pcut and better than the Naive Bayes approach examined by Yang and Liu. The macro-averaged F_1 of 51.18% is better than when either Pcut or density estimation is used alone, beating three of the five methods tested by Yang and Liu.

4.4 Research Related to Density Estimation

Joachims (1997) has implemented a probabilistic version of a Rocchio-based text categorization system that uses TF*IDF representations of documents and categories to compute probabilities of categories given a test document. The method that system uses appears quite different from standard Rocchio, but Joachims shows that the two are similar and that they would be equivalent under certain assumptions. My system applies standard Rocchio first, and then applies density estimation to convert similarity scores to probabilities. Joachims tests his system on two representative categories of the Reuters corpus[3], whereas I test my system on 90 Reuters categories representing the common benchmark mentioned in Section 4.2 and described in Section 2.8. Joachims compares his probabilistic TF*IDF system to a standard version and Naive Bayes, whereas I compare my system against all methods tested by Yang and Liu (1999) for Reuters categorization, and against several systems that comprise the publicly available Rainbow package (results of these systems have already been presented in Section 3.2) for categorization of the data sets that comprise the corpus I have created (as described in Section 3.1).

The concept of converting category similarity scores to probabilities (or recalibrating category probabilities) is not new. Some recent research exploring this potential is discussed in two recent papers by Bennett. Bennett (2000) provides evidence that Naive Bayes systems typically produce probabilities close to 0 or 1, a problem that I have noticed in my own research experience but have never formally analyzed, and he begins to discuss methods that could be used to recalibrate the probabilities. In a later paper (Bennett, 2002), he introduces methods of fitting asymmetric Gaussian and Laplace distributions to the output of a Naive Bayes classifier and a linear SVM classifier. The methods described in these papers are designed to work for binary categorization tasks involving separate YES/NO decisions for all document/category pairs.

In (Sable and Hatzivassiloglou, 2000), my co-author and I have reported detailed results of classifying images based on associated text as either *Indoor* or *Outdoor*. Even then, my system, which relied on a combination of Rocchio/TF*IDF and a much simpler version of density estimation, outperformed a competing image based system (Szummer and Picard, 1998) that was optimized to perform the same task. In another paper (Paek et al., 1999), we have reported the results of integrat-

[3]Joachims also tests his system on the 20 Newsgroups collection described in Section 2.8.

ing my system at that time with an image based system. I have since improved my system using the generalized method of density estimation described in this chapter and in (Sable, McKeown, and Hatzivassiloglou, 2002). The previous method could only be applied when there are exactly two mutually exclusive categories, but the current version can be applied when dealing with any number of mutually exclusive categories, or when dealing with a multi-label, binary categorization task. In addition, the current method leads to significantly better results even for the task involving exactly two mutually exclusive categories.

4.5 Concluding Discussion of Density Estimation

This chapter presents a novel application of an established mathematical technique, density estimation, that significantly improves the performance of my Rocchio-based text categorization system for two out of three experiments. This technique can be applied to the results of any system that categorizes documents by computing a similarity score for every category. The results of this chapter suggest that the use of density estimation should be considered for categorization tasks; experiments within the training set can be used to determine whether it is likely to improve the performance of a system.

In addition to improving performance, density estimation provides probabilistic confidence measures for a system's predictions, whereas a standard Rocchio-based system (and many other alternatives to which density estimation could be applied) can only provide category rankings or YES/NO decisions. An analysis of the validity of these confidence measures for the first two experiments discussed in this chapter (those involving the *Indoor* versus *Outdoor* data set and the Events data set) is provided in Appendix H. Future work involving density estimation might involve using the confidence measures to combine results of various systems for a task. If multiple systems applied to the same task suggest different outcomes, the probability estimates provided by the density estimation technique represent levels of confidence of each prediction. Each individual prediction could be weighted according to the estimate; this would be similar to the weighted linear combination strategy for combining systems that is mentioned in Section 7.5. Alternatively, the most confident prediction for each new document can be used; this would be similar to the dynamic classifier selection strategy for combining systems that is also mentioned in Section 7.5, although with density estimation, there is no direct comparison of new documents to training set documents. In Chapter 7, I discuss combining high-precision/low-recall rules with other systems, and in Section 7.6, I mention the possibility of using density estimation as a means of combining these rules in a potentially more helpful manner. In Chapter 8, I discuss one method of combining image features and text for the categorization of images, and in Section 8.3.3, I outline another procedure involving density estimation that might work better. Although I have not yet carried out such lines of research, it is clear that density estimation may be useful in ways other than simply improving the performance of individual systems.

Chapter 5

Bins

As I have explained in Section 2.4, bag of words approaches to text categorization often represent documents as vectors of weighted words (or terms). Word weights are usually computed by combining separate features in some fashion, for example, by multiplying together the term frequency (TF) and inverse document frequency (IDF), as described in Section 2.4.2. I have also explained that the Naive Bayes method of text categorization (Lewis, 1998) empirically estimates term weights for each individual word (or term) that appears in the training set based on estimated probabilities of seeing each word in a document of each possible category. This method is prone to inaccurate term weights for words that occur infrequently in the training set. Words that have never been seen in the training set are ignored since all of the estimated probabilities are zero, and words that appear in only one category in the training set might appear to give that category infinite likelihood if they appear in a document from the test set. These observations have led to my next attempt to improve a popular text categorization approach, in this case Naive Bayes. In this chapter I show that empirically estimating term weights for bins, where each "bin" contains a group of "similar" words, instead of estimating weights for individual words, avoids the pitfalls associated with scarce evidence.

This particular use of bins can be thought of as a smoothing technique added to Naive Bayes. In addition to avoiding inaccurate term weights, the use of bins provides evidence indicating which features of words are most important for indicating categories.

Using bins, I have implemented a user-friendly text categorization system called BINS, and it is shown in this chapter and throughout the rest of this thesis that this system is extremely competitive with the state-of-the-art.[1] The BINS system is ready to be made publicly available for research purposes; once this happens, I will post instructions describing how to obtain the system at http://www.cs.columbia.edu/~sable/bins.html. In addition to using bins for text categorization, BINS also allows the user to combine the bin-based methodology with standard Naive Bayes. The idea is that the term weight for an individual word can be used when there is enough evidence, and otherwise we can fall back to the term weight for the word's bin. It is also possible to combine the two weights. I show in this chapter that, for my data sets, bins alone performs better than Naive Bayes alone, and a combination of the two performs better than either individually.

5.1 Research Related to Bins

The Speech Recognition literature has developed a number of methods for smoothing term frequencies (e.g., Chapter 15 of (Jelinek, 1998)). These methods are important when the raw counts are small, and particularly important when the counts are zero. Both the Good-Turing method and the Deleted Interpolation method estimate $r*$, an adjusted value of r, where r is the frequency with which the term t appears in one corpus, and $r*$ is the frequency with which t is expected to appear in another corpus of similar size and material.

[1] The creation of BINS and the research discussed in this chapter have been performed in collaboration with Kenneth W. Church at AT&T.

The Deleted Interpolation method assigns each term, t, to a bin, b, usually based on the frequency r, but binning rules can also make use of other variables. Section 15.6 of (Jelinek, 1998), for example, discusses so-called "enhanced" binning rules where bigrams are assigned to bins not only based on their joint frequencies but also the frequencies of their parts. In my work, terms are assigned to bins based on statistics that are commonly used in the text categorization literature.

The Deleted Interpolation method splits the training collection into two pieces. The first piece is used to assign terms to bins, and to compute the number of terms that have been assigned to each bin, n_b. The second piece is used for calibrating bins. x_b is the number of times that the terms in bin b are found in the second piece. The final answer is then $r* \approx x_b/n_b$. In general, $r*$ tends to be slightly smaller than r in most cases except when $r = 0$. The adjustments are important when r is small, especially when $r = 0$. All of the terms in a bin receive the same adjusted frequency, $r*$.

(Umemura and Church, 2000) shows how the Deleted Interpolation approach can be generalized to estimate likelihood ratios instead of frequencies in an information retrieval application. In this chapter, I discuss a similar approach that I use for text categorization. Text categorization is interestingly different from information retrieval because there tends to be relatively more supervised training material.

5.2 Data Sets Used for Experiments with Bins

In general, my research has focused on news documents and their embedded images. For my experiments with BINS, the bin-based system that I implemented, I have used the same data sets as for the first two experiments with my density estimation system, as discussed in Section 4.2. The first experiment involves the *Indoor* versus *Outdoor* data set, first described in Section 3.1.2.2, and the second experiment involves the Events data set, first described in section 3.1.2.3. I am not be reporting results of using bins for the Reuters data set, since the current version of BINS is designed to work only for tasks involving mutually exclusive categories. You can see results for an older version of my system applied to a Reuters data set as published in (Sable and Church, 2001).

5.3 Bins and Features

The main principle behind a bin-based system is to group words with similar features into a common bin. Features should be chosen such that words with the same values of these features would be expected to have very similar values of the term weights being estimated. When dealing with text categorization, once words have been placed into bins, a bin-based system estimates the likelihood of a word from a specific bin appearing in a document of a specific category with a specific occurrence count. It is therefore important to group words together in such a way that the words in a single bin are expected to occur with about equal probability in a document of a specific category.

The current version of my BINS system *creates a separate set of bins for each category.* The reason that there is a separate set of bins for each category deals with the term weights being estimated; each term weight is an estimated probability that a word from a bin will appear in a document *of a specified category.* This is analogous to the probabilities estimated for individual words with a standard Naive Bayes system. In order to create a single set of bins for all categories, we would have to somehow group words together such that each set contained only words that had the same likelihood as each other for all categories; that would be difficult, and the bins would not necessarily be very meaningful.

BINS uses two features (by default) to group words together in bins. The first such feature is what I call the "category count" for the category associated with the given bin (remember from the previous paragraph that BINS creates a separate set of bins for each category). This is defined as the number of training documents of the specified category in the first half of the training set that contain the word. This should intuitively seem like a good feature to use. If, for example, a group of words all appear in k out of n documents labeled as category c in the first half of the training set, it seems likely that they will all occur with approximately the same frequency as each other in later documents that also belong to category c. (You might be tempted to further assume that this frequency should be estimated as k/n, but the entire point of bins is that this is not necessarily a good estimate when k is small, especially when k is 0. See the discussion of Deleted Interpolation in Section 5.1, or the Scientific American article by Efron and Morris (1977) explaining Stein's Paradox, which dictates that this intuitive guess is often not the best prediction.)

The second feature used to group words together in bins is the quantized IDF of the word, based on the first half of the training set. IDF, or inverse document frequency, is a very common statistic used in the text categorization and other IR literature, as described in Section 2.4.2. BINS quantizes the value simply by truncating it; if these values were not quantized, there would be too many possible values, and the bins would be too small, defeating the purpose of binning in the first place. The reason that the IDFs are based only on the first half of the training set is that this is the portion of the training set that is used to place words into bins. Calculating the IDF of a word based on the entire training set would be problematic, since the IDF itself would then give an unfair hint as to how frequently the word appears in the second half of the training set; I explain in Section 5.4.1 that the second half of the training set is used to estimate the likelihood of seeing words from each bin in a document of a specified category. Using IDF as a binning feature may seem less intuitive than using category counts; however, it turns out that these values are helpful, and this is discussed in more detail in Section 5.4.3.

Another option for computing IDFs would be to base the values on a larger, different corpus. On the one hand, there would be more data, but on the other hand, the data would not be as representative of the test set. Some experiments that I have conducted suggest that using a larger corpus to compute IDFs might be beneficial for some tasks, but for other tasks it causes problems; In any case, I have decided that I want my BINS system to be general-purpose, without needing to rely on outside information. Still, I discuss this option in more detail in Appendix K. In summary, the use of IDFs based on a much larger corpus slightly improves performance for the *Indoor* versus *Outdoor* data set and the Events data set when bin weights are always used, but when bin weights are used in conjunction with Naive Bayes weights (as is discussed later in the chapter in Section 5.7), performance stays the same or degrades. The appendix also describes why for certain tasks, such as Reuters categorization, the use of weights computed based on another corpus is dangerous.

Intuition	Word	Indoor Category Count	Outdoor Category Count	Quantized IDF
Clearly Indoor	conference	14	1	4
	bed	1	0	8
Clearly Outdoor	plane	0	9	5
	earthquake	0	4	6
Unclear	speech	2	2	6
	ceremony	3	8	5

Table 5.1: Values of binning features are used to assign words to bins.

Table 5.1 shows the values of the features for six selected words when assigning words to bins based on the first half of the training set for the *Indoor* versus *Outdoor* data set. Based on these features, every word in the corpus is assigned to a bin corresponding to each category. For example, the word "plane" is assigned to an *Indoor* bin consisting of all words with a quantized (truncated) IDF of 5 that appear in zero *Indoor* documents in the first half of the training set, and it is assigned to an *Outdoor* bin consisting of all words with a quantized IDF of 5 that appear in nine *Outdoor* documents in the first half of the training set. In this particular case, there may be enough evidence to predict the probability of seeing the word "plane" in a future *Outdoor* document just based on the word alone, but there is clearly not enough evidence to predict the probability of seeing the word "plane" in a future *Indoor* document. However, even if the word "plane" never occurs in an *Indoor* document in the entire training set (both halves), chances are that at least one word in the same *Indoor* bin as "plane" occurs at least once in an *Indoor* document in the second half of the training set; this then leads to a non-zero probability which is likely a better estimate. In this way, binning can provide credible term weights, even for words that never occur in a category.

BINS also allows other features to be used in order to group words into BINS. One such feature is burstiness, as suggested by (Umemura and Church, 2000), which was used by default in the earlier version of the system described in (Sable and Church, 2001) and is discussed in Appendix I. Another is an alternative to IDF of my own invention that I call *shared scaled category likelihoods*, which is described in Appendix J. In both of these cases, using these weights in addition to (or instead of) IDF to bin words has inconclusive results when applied to the data sets used in the experiments discussed in this chapter; in other words, some results improve while others degrade, and in general, the changes are small. The appendices explain my decisions not to use these features by default. Finally, BINS also allows the user to provide any arbitrary weights for words that the user wishes to specify (by providing a file that maps each word to a weight). However, the experiments discussed in the remainder of this chapter all use only the default features (category counts and quantized IDFs) to group words into bins.

5.4 Calculating Term Weights

5.4.1 Methodology for Calculating Term Weights

Once bins have been determined based on the first half of the training set, the second half of the training set is used to empirically estimate term weights for all bins. Each bin is assigned multiple term weights corresponding to different possible occurrence counts, or term frequencies, of words in documents (this will be elaborated shortly). Given a bin related to a specific category, we estimate the probability that a word belonging to the bin will appear in an article of the same category with some specific count. The log of this probability is used as a term weight for the bin.

BINS estimates separate term weights for various possible occurrence counts. For each bin, the occurrence counts considered are, by default, zero, one, two, three, or four or more. All occurrence counts above the maximum are truncated to the maximum, and the user has the option of changing this maximum, but the experiments discussed in this chapter all use the default of four (determined empirically based on some initial experiments). For example, consider a word such as "earthquake", and its *Disaster* bin for the experiment with the Events categories; separate term weights are estimated according the the probabilities of seeing the word "earthquake" in a *Disaster* document one or more times, two or more times, three or more times, and four or more times. It is expected that, other things being equal, the more a word occurs in a document, the more indicative it is of a category.

More formally, let $W_{bin(c,w)}$ represent the set of all words in the same bin as word w corresponding to category c. This set of words is determined based on the first half of the training set. Let D_c be the set of all documents belonging to category c in the second half of the training set. For any document d and word w, let $I(d, w, i)$ be 1 if d contains the word w at least i times and 0 otherwise.

Then, we can estimate the probability of observing a word w at least i times in a document of category c as being:

$$P(w,i|c) = \frac{1}{|W_{bin(c,w)}|} \times \sum_{w_j \in W_{bin(c,w)}} \left[\frac{1}{|D_c|} \times \sum_{d_k \in D_c} I(d_k, w_j, i) \right] \tag{5.1}$$

The part in brackets represents the estimated probability, based on the second half of the training set, that a single word (w_j) from a bin will show up at least i times in a document of category c. The expression as a whole simply averages the probabilities for all of the words in the same bin as w corresponding to the specified category. The log of this averaged probability, i.e. $\log_2 P(w,i|c)$, is the term weight for the bin corresponding to the occurrence count of i and the category c. Note that all term weights are zero or negative, since we are taking the logs of probabilities, so values closest to zero represent the most likely scenarios.

5.4.2 An Example of Calculating Term Weights

The above steps are best illustrated by example. As shown in Table 5.1, for the first experiment, the word "plane" appears in nine *Outdoor* documents in the first half of the training set but no *Indoor* documents in the first half of the training set. The quantized IDF of "plane", according to the first half of the training set, is 5. Therefore, "plane" is placed in one bin representing all words with a truncated IDF of 5 that appear in nine *Outdoor* documents, and I will refer to this as the *Outdoor* bin of "plane". The same word is placed in another bin representing all words with a truncated IDF of 5 that appear in zero *Indoor* documents, and I will refer to this as the *Indoor* bin of "plane". When the second half of the training set is examined, it is noted that the estimated probability that a word belonging to the same *Indoor* bin as "plane" appears one or more times in an *Indoor* document is 5.31×10^{-3}, and so the term weight for this bin corresponding to the occurrence count of 1 is:

$$\lambda_{Indoor} = \log_2(5.31 \times 10^{-3}) = -7.56 \tag{5.2}$$

It is also noted that the estimated probability that a word belonging to the same *Outdoor* bin as “plane” appears one or more times in an *Outdoor* document is 2.13×10^{-2}, and so the term weight for this bin corresponding to the occurrence count of 1 is:

$$\lambda_{Outdoor} = \log_2(2.13 \times 10^{-2}) = -5.55 \tag{5.3}$$

The difference between the term weight associated with a word’s *Indoor* bin and the term weight associated with a word’s *Outdoor* bin is a log likelihood ratio comparing the likelihood of seeing the word (or one with the same features) in one category versus the other. In this case, $\lambda_{Indoor} - \lambda_{Outdoor} = -7.56 - (-5.55) = -2.01$, which means that if a word with the same features as “plane” is seen in a document, it is about $2^{2.01} = 4.03$ times more likely to be an *Outdoor* document than an *Indoor* document.

This example should help to illustrate why the training set is divided into two halves. The first half of the training set determines bins, and the second half is used to calibrate bin probabilities. Let’s say that “plane” does not occur in any *Indoor* documents in the entire training set. This does not mean that if the word is seen in a test document, it is necessarily an *Outdoor* document. Since “plane” is placed in an *Indoor* bin with other words that occur in no *Indoor* documents in the first half of the training set, and some of these words appear in some *Indoor* documents in the second half of the training set, we are able to estimate a probability for the bin as a whole of seeing a word from the bin in an *Indoor* document.

Table 5.2 shows the results of subtracting the *Outdoor* term weights from the *Indoor* term weights for the bins of the the same six words shown in Table 5.1. As expected, words such as “conference” and “bed” are good *Indoor* indicators, while words such as “plane” and “earthquake” are good *Outdoor* indicators. The

Intuition	Word	Indoor Category Count	Outdoor Category Count	Quantized IDF	$\lambda_{Indoor} - \lambda_{Outdoor}$
Clearly Indoor	conference	14	1	4	4.84
	bed	1	0	8	1.35
Clearly Outdoor	plane	0	9	5	-2.01
	earthquake	0	4	6	-1.00
Unclear	speech	2	2	6	0.84
	ceremony	3	8	5	-0.50

Table 5.2: Term weights for these words fit intuition as to which words indicate which category; that is, the final column contains large positive values for clearly *Indoor* words, large negative values for clearly *Outdoor* words, and values closer to zero for words that are unclear.

words "speech" and "ceremony" each show a slight preference for one category over the other, but according to term weights, each is less than twice as likely in the favored category.

5.4.3 Justification of Using IDF as a Binning Feature

It was explained in Section 5.3 that category count should intuitively seem like a good feature to use. The same, however, should not be said for IDF. Let's say, for instance, we are dealing with the Events categories, and we know the category count of a word in a category; perhaps, for example, we are dealing with a word that occurs zero times in the first half of the training set in the *Disaster* category. The IDF, on the other hand, actually represents the frequency (or rarity) of the word in the entire first half of the training set, and not just in the specified category (see Section 2.4.2 for an explanation of IDF). In combination with the category count for a specified category, then, the additional information provided by the IDF is the word's level of frequency in the rest of the first half of the training set; but, is there any reason to believe that the frequency of a word in documents of other categories should have any influence in the prediction of the future frequency of the word in the specified category? It turns out that, according to statistics, the answer appears to be yes.

IDF	$\lambda_{Disaster}$ (category count $= 0$)	$\lambda_{Disaster}$ (category count $= 1$)
1	-5.70	-4.39
2	-5.49	-5.70
3	-6.70	-5.73
4	-7.26	-6.20
5	-7.74	-6.40
6	-8.17	-6.65
7	-8.85	-7.23
8	-10.05	-7.84

Table 5.3: The IDF significantly influences the predictions of the likelihood of seeing a word in a future document of some specified category.

Table 5.3 shows the lambdas computed for various bins for the second experiment (dealing with the Events categories) corresponding to occurrence counts of one or more. The first column shows values computed for the *Disaster* bins of words that did not occur in any *Disaster* documents in the first half of the training set, and the second column shows values computed for the *Disaster* bins of words that occurred in exactly one Disaster document in the first half of the training set. (Remember that binning is most important for bins corresponding to low category counts, since these represent cases with scarce evidence.) Each row shows the lambdas computed for a different IDF. Note that as IDF varies holding everything else the same, the lambdas vary as well, perhaps more than you would expect. These lambdas are logs of probabilities (using a base of two), and so this table shows, for example, that a word that never occurs in a *Disaster* document in the first half of the training set but is common (low IDF) in the rest of the first half of the training set (in documents that belong to other categories) is about 20 times as likely to show up in a future *Disaster* document compared to a word that never occurs in a *Disaster* document in the first half of the training set and is also rare in the rest of the first half of the training set.

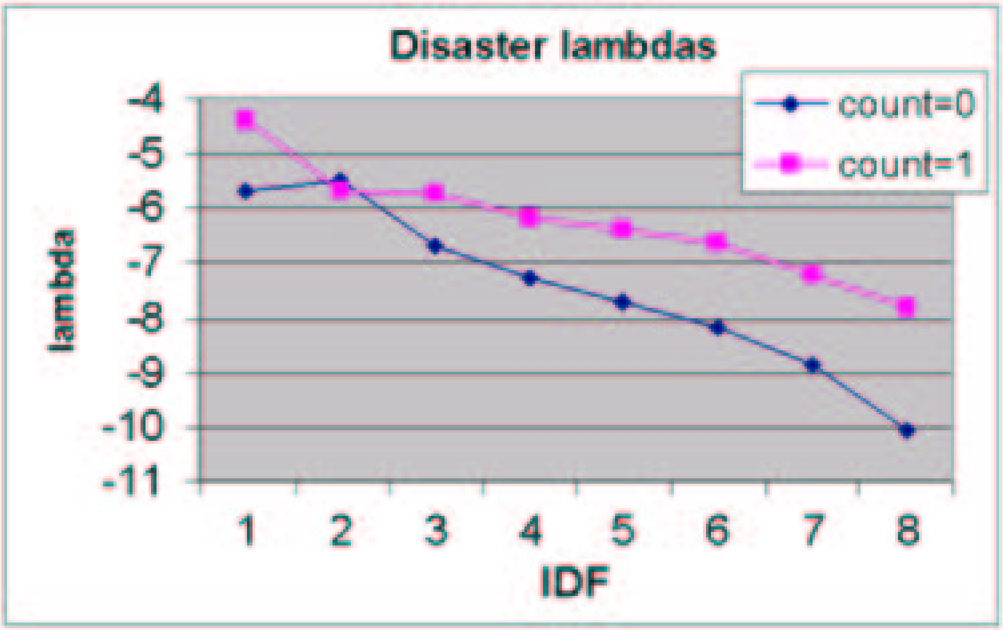


Figure 5.1: As IDF increases, holding category count constant, the likelihood of seeing a word in a document of a specified category decreases.

Figure 5.1 shows the same information as Table 5.3 in graphical format. The graph makes it clear that lambdas (which represent the estimated likelihood of seeing a word in a document of the specified category) decrease as IDF increases, even holding category count constant. It appears that, everything else being equal, there is approximately an inverse, linear relationship between IDF and lambda. Also, as is to be expected, when holding IDF constant, in almost all cases, the estimated likelihood of seeing a word with a category count of 1 is higher than the estimated likelihood of seeing a word with a category count of 0 for the category. The only small anomaly in the graph is that the lambda for a category count of 0 and an IDF of 2 seems slightly higher than it should be; however, the corresponding bin is very small (since there are very few words that occur so commonly in the corpus as a whole to achieve such a low IDF but never appear in a *Disaster* document in the first half of the training set), so this lambda is still based on scarce evidence and may not be accurate. In any case, we are still likely better off using the bin weight than that of individual words contributing to the bin.

It should be noted that just because IDF seems to be correlated, and therefore indicative, of the future probability that a word will show up in a future document of a specific category, this does not necessarily mean that performance improves when it is used as a binning feature. Every new binning feature makes bins smaller, and therefore less accurate. There is a performance gain only if the benefit of using an additional indicative feature outweighs the negative effect of making the bins smaller. Appendix L compares the use category counts alone to the use of category counts with IDF for the *Indoor* versus *Outdoor* data set and the Events data set. It turns out that IDF does seem to improve performance for the Events data set, but it leads to a small degradation in performance for the *Indoor* versus *Outdoor* data set. Still, based on the justification presented in this section and the vast use of this feature in the text categorization literature, I have decided to have the system use the feature by default.

5.5 Overall Methodology of BINS

BINS provides simple tools that allow the user to divide the training set into two halves. If the user wishes, he or she does not have to make the two "halves" actually have equal size, but this is what I recommend. Once the training set is split, a single script is provided that performs all training, testing, and even evaluation (if there are known labels for the documents in the test set). If the user wishes, however, he or she may do training and testing separately; then, once the system is trained, it can be applied to various test sets without retraining.

The methodology of training has been fully described in Sections 5.3 and 5.4. The first half of the training set is used to place words into bins, and then the second half of the training set is used to calculate term weights (estimated likelihoods) for each bin. Once term weights for all bins have been calculated, BINS loops through the test documents to predict categories. For each test document, BINS iterates through its words, summing the appropriate term weights for each category. For example, for the second experiment, five sums are generated, one for each category. The first is the sum of the term weights for the *Struggle* bins associated with all words in the document (taking into account the specific occurrence counts of the words), the second is the sum of the term weights for the *Politics* bins, etc. Whichever category has the highest sum is considered the most likely and is therefore predicted. The likelihood of bins are assumed to be independent. This is similar to the independence assumption of words in Naive Bayes, and in fact the two methods are virtually identical if every word is assigned its own bin. The use of bins in this fashion can be considered to be a smoothing technique applied to Naive Bayes.

Figure 5.2 shows pseudo-code summarizing the algorithm. This pseudo code assumes that everything is run in default mode, and some things are left out (for instance, IDFs actually need to be computed for each word), but the main points are here. In actuality, BINS consists of a collection of Perl scripts with a total of approximately 3,500 lines of code. Much of this code, however, deals with options that are not discussed in this chapter. Some of these options are discussed in Appendices I, J, K, L, M, and N. The first part of the algorithm (corresponding to lines 1 through 16 of the pseudo-code) determines the features associated with all words based on the first half of the training set, maps words to bins accordingly, and computes the size of each bin. The next part of the algorithm (corresponding to lines 17 through 44 of the pseudo-code), uses the second half of the training

```
 1 # Determine category counts of words in corpus.
 2 for every document in first half of training set {
 3     current_category <-- category[document];
 4     for every word in document {
 5         count[word, current_category] += 1;
 6     }
 7 }
 8 # Determine bins for all known words.
 9 for every known word {
10     for every category {
11         category_count <-- count[word, category];
12         current_bin <-- bin(IDF[word], category_count);
13         # Increment size of current bin.
14         size[current_bin] += 1;
15     }
16 }
17 # Determine count of every (bin, occurrence count) pair.
18 for every document in second half of training set {
19     current_category <-- category[document];
20     for every word in document {
21         category_count <-- count[word, current_category];
22         current_bin <-- bin(IDF[word], category_count);
23         occurrence_count <-- term_frequency[word, document];
24         # Increment count for this word's (current_bin, occurrence count) pair.
25         count[current_bin, occurrence_count] += 1;
26     }
27     # Fix counts for occurrence counts of 0 (uses bin sizes) here.
28 }
29 # Estimate probability of every (bin, occurrence count) pair.
30 for every bin {
31     total = 0;
32     for every possible occurrence_count (0 to MAX) {
33         total += count[bin, occurrence_count];
34     }
35     for every possible occurrence_count (0 to MAX) {
36         probability[bin, occurrence_count] = count[bin, occurrence_count] / total;
37     }
38 }
39 # Calculate term weights.
40 for every bin {
41     for every possible occurrence_count (0 to MAX) {
42         lambda[bin, occurrence_count] = log (probability[bin, occurrence_count]);
43     }
44 }
45 # Loop through test set.
46 for every document in test set {
47     for every possible category {
48         score[category] = 0;
49     }
50     for every word in document {
51         occurrence_count <-- term_frequency[word, document];
52         for every possible category {
53             category_count <-- count[word, category];
54             current_bin <-- bin(IDF[word], category_count);
55             score[category] += lambda[current_bin, occurrence_count];
56         }
57     }
58     # Assign document to category with highest score here.
59 }
60 # Various results are computed and displayed here.
```

Figure 5.2: This pseudo-code represents the algorithm used to conduct an entire experiment, including training (lines 1 - 44) and testing (lines 45 - 60).

set to estimate probabilities of all possible (bin, occurrence count) pairs, and corresponding term weights are computed. Some extra code is needed to compute counts for all instances of occurrence counts of 0 (see the comment at line 27 of the pseudo-code), and there are also checks throughout the actual code to avoid infinities. Finally (corresponding to lines 46 through 60 of the pseudo-code), the test set is examined. For every test document, every word is mapped to its appropriate bins (one for each category), and the score for each category is incremented by the appropriate term weight. Each test document is assigned to the category with the highest score. On typical Unix systems, the actual system can complete a full run including training and testing for the *Indoor* versus *Outdoor* data set in a few seconds, and it can complete a full run for the Events data set in about one minute. Both data sets consist of approximately 900 training documents and 450 test documents; the *Indoor* versus *Outdoor* experiments are quicker because only first sentences of captions are used, whereas the Events experiments involve lengthy news articles.

I have found that one way to generally improve results of my system is to repeat training and testing twice, swapping the two halves of the training set in between runs. So the second time around, the second half of the training set is used for mapping words to bins, and the first half is used to determine term weights. This way, if a word happens to occur in documents of a specific category more in one half than the other, the term weights used for the word (based on the two bins the word is mapped to in the two separate runs) should balance each other out. Therefore, BINS also provides a script that can combine the results of two runs, and a final script that does everything at once (training and testing twice swapping the two halves of the training set in between, then combining and analyzing results). The results discussed in the next section all are based on experiments that train and test twice in this manner.

5.6 Results and Evaluation of Experiments with Bins

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
My systems			
BINS	85.8	75.1	90.1
Naive Bayes	84.5	70.9	89.4
Rocchio/TF*IDF	80.7	69.9	85.7
Density Estimation	86.1	73.7	90.5
K-Nearest Neighbor	82.7	65.8	88.4
Rainbow systems			
Naive Bayes	85.4	73.5	89.9
Rocchio/TF*IDF	84.5	73.2	89.1
K-Nearest Neighbor	77.8	65.3	83.6
Probabilistic Indexing	86.3	78.1	90.0
Support Vector Machines	82.0	66.9	87.7
Maximum Entropy	84.5	70.9	89.4

Table 5.4: BINS performs better than Naive Bayes and most other systems tested for the *Indoor* versus *Outdoor* data set.

Table 5.4 shows the results of BINS and several other systems tested for the first experiment (involving the *Indoor* versus *Outdoor* categories). These include all of the systems shown in Table 4.3 in the previous chapter, the BINS system described here, and also my own implementation of a Naive Bayes system. The most important comparison is that of my BINS system against my own Naive Bayes system, which is identical to BINS except that term weights are computed for individual words; this is the only comparison that demonstrates the effect of using bins.[2] BINS beats both versions of Naive Bayes, and also performs better than

[2]There are many reasons that my Naive Bayes system (or other systems I have developed) would not achieve exactly the same result as a Rainbow (or other) system using the same general methodology. Differences might include tokenization rules (i.e. what counts as a word - my system simply uses all strings of alphabet letters in the text, converting all letters to lower case),

most of the other systems tested (the exceptions being my own density estimation system described in Chapter 4 and Rainbow's Probabilistic Indexing system). At this level, BINS is performing significantly better (according to a one-sided χ^2 test) than the bottom four systems; the other systems fall within a 95% confidence interval. Further improvement is demonstrated in Section 5.7, which discusses the combination of bin weights and Naive Bayes weights, that pushes BINS performance to a level above that of all other systems tested. As reported in the previous chapter (see Section 4.3), a a baseline classifier that picks the larger category every time (*Outdoor*) achieves a 71.2% overall accuracy, which is significantly lower than all the automatic systems tested; humans who are asked to predict labels after being shown only captions achieve an 87.6% overall accuracy, which I consider to be a reasonable upper bound for how well a text categorization system might be expected to do.

Table 5.5 shows the results of BINS and all other systems tested for the second experiment (involving the Events categories), which are the same as those listed in the previous table. Once again, BINS performs better than Naive Bayes and most other systems tested. In both experiments, BINS not only improves upon the overall accuracy of the similar Naive Bayes implementation, but also improves performance for every category. Only two of the systems tested outperform BINS on this data set, those being Rainbow's SVM system and Rainbow's Maximum Entropy system. At this level, BINS is performing significantly better (according to a one-sided χ^2 test) than the bottom three systems; the other systems fall within a 95% confidence interval. Further improvement is demonstrated in Section 5.7, which discusses the combination of bin weights and Naive Bayes weights, that pushes

whether or not stop words are automatically removed (my system does not do this), how to handle occurrence counts (I believe Rainbow simply multiplies the term weight for a single occurrence by the occurrence count, whereas my technique has been described previously in the chapter), etc. In this case, my own Naive Bayes system performs slightly worse for this particular data set than that which is part of the Rainbow package; some of the differences probably help, but others hurt, and the net effect happens to be negative for this case.

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
My systems						
BINS	88.0	87.5	88.3	97.2	83.1	60.0
Naive Bayes	86.0	85.4	87.2	96.7	81.4	27.6
Rocchio/TF*IDF	87.1	85.0	88.4	98.8	79.2	60.0
Density Estimation	84.9	83.7	86.0	97.3	80.0	34.3
K-Nearest Neighbor	84.0	81.1	82.1	93.9	81.3	65.0
Rainbow systems						
Naive Bayes	87.6	86.2	86.3	96.7	89.1	61.5
Rocchio/TF*IDF	87.4	81.1	85.3	97.7	88.4	68.3
K-Nearest Neighbor	81.9	80.0	79.7	95.6	75.6	63.2
Probabilistic Indexing	86.5	83.6	84.8	97.2	89.4	65.0
Support Vector Machines	88.7	88.1	89.2	96.2	87.0	57.9
Maximum Entropy	88.3	88.1	87.9	95.7	87.9	55.6

Table 5.5: BINS performs better than Naive Bayes and most other systems tested for the Events data set.

BINS overall accuracy for these categories above 90%, thus beating all of the other systems tested. A baseline categorizer that picks the largest category every time (which happens to be *Struggle*, based on the training set) would achieve only a 30.5% overall accuracy.

5.7 Combining Bins and Standard Naive Bayes

There is an interesting trade-off when using bins instead of standard Naive Bayes. On the one hand, it is expected that term weights for bins are more accurate, since they are based on multiple words and have additional evidence compared to single words. On the other hand, half of the training set is "lost" (i.e. used to group words into bins), and therefore only half is used to actually estimate the term weights. Although the training/testing is repeated twice, as described in Section 5.5, swapping the two halves of the training set in between, it is not necessarily the case that the two runs balance out in such a way that term weights are as accurate for words with ample evidence as a single term weight based on entire training set for such words. BINS therefore allows the user to combine the binning-approach described in this chapter with standard Naive Bayes. The idea here is to use the term weights for individual words when there is enough evidence, and to back off to the term weight for the word's bin otherwise. BINS also allows the two weights to be combined with various proportions for various levels of evidence.

In order to determine what might be the best way to use the two weights for various levels of evidence, I have divided the original training set into three subsets. The first two subsets are used as a new training set to estimate term weights, and the the third is used to assess the accuracy of these term weights. When computing term weights for the binning approach, the first two subsets of the training set are used separately, one to assign words to bins and the other to estimate term weights, as described in the rest of this chapter. To estimate Naive Bayes term weights, the first two subsets are used together. In addition to assessing the accuracy of individual term weights, I also assess the accuracy of various weighted combinations of the two term weights for various levels of evidence. By "level of evidence", I am referring to the number of documents of a specified category in which a word appears in the training portion of the data. It is expected that the higher this number, the more accurate a Naive Bayes estimate for the word is likely to be, and the less necessary it is to back off to the bin. The BINS system provides a very simple mechanism for the user to indicate how he or she wishes weights to be combined for various levels of evidence.

I have used two methods to assess the accuracy of term weights. The first is entropy, which, roughly speaking, refers to the number of bits needed to indicate the documents that each word appears in. This is averaged for words with each possible level of evidence. Such an entropy is computed for bin term weights, Naive

Bayes term weights, and various combinations of the two term weights. I have considered 101 possible combinations of term weights. For all integers n ranging from 0 to 100, I have tested what happens if the Naive Bayes term weight is used for $n\%$ of the combined weight and the bin term weight is used for $(100 - n)\%$ of the term weight. For each level of evidence, I then record which n leads to the best result (the lowest entropy). Results are extremely similar for both data sets. For words that do not appear at all in a specified category in the training set (level of evidence equals 0), the optimal n is 0, meaning that it is best to rely exclusively on the bin term weight. For words that appear once in a specified category in the training set (level of evidence equals 1), the optimal n is very close to 50, meaning that it is best to average the bin weight and the Naive Bayes weight. For words that appear in two or more documents of a specified category in the training set (level of evidence greater than or equal to 2), n is always very close to 100, meaning that it is best to rely exclusively on the Naive Bayes weight. I will refer to this combination of weights as COMBO #1.

The second method I have used to assess the accuracy of term weights is to determine what combination of weights (what value of n as described in the previous paragraph) minimizes the squared errors for estimated word probabilities at various levels of evidence. This time, the trend is more gradual, especially for the *Indoor* versus *Outdoor* data set. This has led me to also try a second set of weight combinations. For words that do not appear at all in a specified category in the training set, I have BINS rely exclusively on the bin weight; for words that appear in one document of a specified category in the training set, I have BINS use 25% of the Naive Bayes weight and 75% of the bin weight ($n = 25$); for words that appear in two documents of a specified category in the training set, the two weights are averaged equally ($n = 50$); for words that appear in three documents of a specified category in the training set, I have BINS use 75% of the Naive Bayes weight and 25% of the bin weight ($n = 75$); and for words that appear in four or more documents of a specified category, I have BINS rely exclusively on the Naive Bayes weight. This combination of weights will be referred to as COMBO #2. COMBO #2 is more in line with my original intuition of what would work the best.

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
My systems			
BINS (COMBO #2)	87.2	78.0	91.0
BINS (COMBO #1)	86.1	76.2	90.2
BINS (always use bin)	85.8	75.1	90.1
Naive Bayes	84.5	70.9	89.4
Rocchio/TF*IDF	80.7	69.9	85.7
Density Estimation	86.1	73.7	90.5
K-Nearest Neighbor	82.7	65.8	88.4
Rainbow systems			
Naive Bayes	85.4	73.5	89.9
Rocchio/TF*IDF	84.5	73.2	89.1
K-Nearest Neighbor	77.8	65.3	83.6
Probabilistic Indexing	86.3	78.1	90.0
Support Vector Machines	82.0	66.9	87.7
Maximum Entropy	84.5	70.9	89.4

Table 5.6: Combining the bin-based methodology with standard Naive Bayes leads to the best results seen for the *Indoor* versus *Outdoor* data set.

Tables 5.6 and 5.7 add to the list of results those obtained using BINS with the combinations of weights described in the previous two paragraphs. As can be seen, both sets of weight combinations improve the performance of BINS, beating both bin weights and Naive Bayes weights alone. For the *Indoor* versus *Outdoor* data set, COMBO #1 is still marginally beat by Rainbow's Probabilistic Indexing system, but COMBO #2 has the best performance of any system tested. In terms of statistical significance, only the four bottom systems show performance below

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
My systems						
BINS (COMBO #2)	90.3	89.5	90.4	98.9	89.1	63.8
BINS (COMBO #1)	90.3	89.7	90.8	98.3	89.4	62.5
BINS (always use bin)	88.0	87.5	88.3	97.2	83.1	60.0
Naive Bayes	86.0	85.4	87.2	96.7	81.4	27.6
Rocchio/TF*IDF	87.1	85.0	88.4	98.8	79.2	60.0
Density Estimation	84.9	83.7	86.0	97.3	80.0	34.3
K-Nearest Neighbor	84.0	81.1	82.1	93.9	81.3	65.0
Rainbow systems						
Naive Bayes	87.6	86.2	86.3	96.7	89.1	61.5
Rocchio/TF*IDF	87.4	81.1	85.3	97.7	88.4	68.3
K-Nearest Neighbor	81.9	80.0	79.7	95.6	75.6	63.2
Probabilistic Indexing	86.5	83.6	84.8	97.2	89.4	65.0
Support Vector Machines	88.7	88.1	89.2	96.2	87.0	57.9
Maximum Entropy	88.3	88.1	87.9	95.7	87.9	55.6

Table 5.7: Combining the bin-based methodology with standard Naive Bayes leads to the best results seen for the Events data set.

the 95% confidence interval (according to a one-sided χ^2 test), but another three systems (including the similar Naive Bayes system) are right near the borderline. For the Events data set, COMBO #1 and COMBO #2 perform equally well as each other (according to overall accuracy) and better than any other system tested; in fact, using either combination of weights pushes the overall accuracy for this data set to over 90% for the first time. With either combination, only the top two competing systems show performance within the 95% confidence interval (according to a one-sided χ^2 test).

5.8 Concluding Discussion of BINS

This chapter describes the use of bins to compute more accurate term weights for words with scarce evidence. The experiments discussed here indicate that a bin-based system is likely to perform better than one that estimates term weights for individual words, and by combining the two approaches, the performance seems to be extremely competitive with all methods I tested. The specific methodology used by my bin-based BINS system can be thought of as the use of a smoothing technique to generalize Naive Bayes.

I believe that there is a lot of future potential for bins and text categorization beyond what is discussed in this chapter. There is likely enough potential in bins that research in this area could constitute a thesis in and of itself. In addition to improving performance by smoothing, bins can be used to determine what features are important for a task. By testing various features, and examining estimated term weights varying one feature while holding the others constant, one can determine if the feature is likely to make a difference according to the training data. My BINS system allows the use of certain additional features that are not used by default; two of these are discussed in Appendices I and J. Many other features are possible, and BINS also allows users to create their own features and specify them for binning; for example, see Appendix K which explores the use of IDFs based on a much larger, alternative corpus instead of the training set. Appendix L explores the use of category counts alone to bin words. Appendix M summarizes the other appendices mentioned in this paragraph.

The use of bins additionally provides a very interesting way to incorporate unlabeled data into training. Because a large amount of unlabeled data is generally easy to obtain, research in utilizing unlabeled data for training has generated a lot of interest (e.g. (Yarowksy, 1995; Blum and Mitchell, 1998; Abney, 2002)). My BINS implementation provides a way to use the data but to weight it less than labeled data using a method described in Appendix N. Although I have not yet seen positive results with this method, I believe it is very promising for reasons explained in the appendix.

Chapter 6

Using NLP

We have seen in Section 2.4 that almost all standard approaches to text categorization rely on *bag of words* approaches; i.e. each text document is represented as a vector of weighted words. Systems generally do not rely on syntax or semantics when computing statistics and making decisions. The use of Natural Language Processing (NLP) to aid text categorization and other IR applications has received a lot of attention, and many believe that there is tremendous potential in this area, but results have been mixed, at best (Strzalkowski, Lin, and Perez-Carballo, 1998; Strzalkowski, 1999; Voorhees, 1993; Smeaton, 1999; Elworthy, 2000). Many of these attempts have been applied to specific tasks involving lengthy textual documents for which standard methods have been performing adequately. In this chapter, I discuss a text categorization task that has come up naturally in the course of my research for which deeper NLP techniques are necessary for optimal performance.

In this thesis, I have been focusing on the categorization of images based on associated text. As has been explained throughout this thesis, the categories applied to images can be quite different than categories applied to full-length text documents such as articles, e-mails, or web pages. The experiments discussed throughout most of this chapter involve the categorization of the images from the

Disaster image data set; that is to say, it concerns the categorization of captioned images that are embedded in news documents concerning disasters, and the possible categories for the images are *Workers Responding*, *Affected People*, *Wreckage*, and *Other*, defined to be mutually exclusive. This data set was not set up with NLP based techniques in mind; rather, I wanted to explore hierarchal categories, which can present new challenges (as explained in Section 2.1). Only after discovering that standard systems do not perform well for this task did I begin to consider the use of NLP based techniques. At the end of the chapter, I also discuss experiments involving images from the *Politics* image data set, for which the possible categories are *Meeting*, *Announcement*, *Politician Photographed*, *Civilians*, *Military*, and *Other*.

Figure 6.1 shows a sample image from the data set along with the first sentence of its caption. This caption contains words that a standard bag of words approach would associate with at least two categories (e.g. "rescuers" $\rightarrow$ *Workers Responding* and "victim" $\rightarrow$ *Affected People*). However, the predicate structure of the sentence emphasizes the rescuers, and this particular image is labeled as a member of the *Workers Responding* category, although you can also see wreckage and a victim within the image. The wreckage and the victim are part of the background, while the rescuers are the object of focus in the foreground. For these categories, the mere mention of a concept in a caption, or the mere presence of an object in an image, is not always enough to indicate a category.

On the other hand, consider an image with a different caption, reading "*A fire victim who perished in a blaze at a Manila disco is carried by Philippine rescuers.*" This caption suggests a focus on the victim as opposed to the rescuers, which implies that the image would be more appropriate for the *Affected People* category. However, the words in the caption are nearly identical. A typical bag of words approach does not have the capacity to distinguish between this hypothetical image

Philippine rescuers carry a fire victim March 19 who perished in a blaze at a Manila disco.

Figure 6.1: The first sentence of this image's caption contains words that a standard bag of words approach would associate with multiple categories.

and the example shown; each word is either present a certain number of times or it is not, and there is no way to capture predicate structure. For certain tasks involving categories such as the ones being discussed here, some linguistic analysis is necessary.

I have shown in Section 3.2.3 that no pre-existing system that I tested is able to perform well for these categories. I eventually became convinced that the main subject and verb of the first sentence of the caption are particularly important in determining the category of an image.[1] These words typically correspond to the object of focus in the image and to what that object is doing. For example, the most helpful words in the caption of the image shown in Figure 6.1 are "rescuers" and "carry". The other words are not helpful, and some, such as "victim", can even be misleading.

This chapter first describes an experiment carried out with human volunteers who viewed captions under varying conditions that supports my hypothesis that consideration of syntax is necessary for optimal performance for this task. It then describes a system I have developed that uses a shallow parser to extract subjects and verbs automatically, together with a novel measure of word-to-word similarity, to place images into these categories. I show that this system outperforms nine competing systems that I have tested for this task.

6.1 Initial Experiments with the Disaster Image Data Set

The creation of the data set discussed in this chapter has been described in Section 3.1.2.4. As a reminder, the final data set consists of 248 images; 98 (39.5%) are

[1]As previously explained, captions typically contain two or three sentences with the first sentence describing the image and the rest giving background information about the related story.

classified as *Workers Responding*, 72 (29.0%) are classified as *Affected People*, 55 (22.2%) are classified as *Wreckage*, and 23 (9.3%) are classified as *Other*. The data set has been randomly divided into a training set and a test set, each containing 124 images.

System	Overall Accuracy %
My systems	
BINS	56.5
Rocchio/TF*DF	57.3
Density Estimation	58.9
Rainbow systems	
Naive Bayes	55.6
Rocchio/TF*IDF	54.0
K-Nearest Neighbor	54.0
Probabilistic Indexing	59.7
Support Vector Machines	54.8
Maximum Entropy	58.1

Table 6.1: The initial results are low for all systems.

My original plan was to use one of my own classifiers to place images into these categories. However, as is the case with standard systems as discussed in Section 3.2.3, it turns out that the performance is not adequate. Table 6.1 shows the results of all systems tested. The performance of the systems ranges from 54.0% to 59.7%. According to a one-sided χ^2 test, the performance of every system falls within the 95% confidence interval of the performance of the best system. Choosing the largest category every time gives a baseline performance of 39.5%. Although all systems are beating the baseline, I did not feel that they are doing as well as possible.

In order to decide in which category to place an image, it is important to determine what is in the image and what that thing is doing. In the sample image shown in Figure 6.1, for example, we see *rescuers carrying*, and that focus of the image places it in the *Workers Responding* category. Words in the caption such as “disco” and “victim” refer to items in the image that are indicative of other categories such as *Wreckage* and *Affected People*, but they do not refer to the focus of the image. I have formed the hypothesis that the main subject and verb of the first sentence of the caption should play a pivotal role in determining an image’s category; if this is correct, it is likely that a system relying on deeper NLP techniques should be able to outperform typical systems for this task. Typical systems relying on bag of words approaches can not account for the predicate argument relationships in the captions.

6.2 Experiments with Humans Viewing the Disaster Image Data Set

To test my hypothesis, I have randomly divided the data set of 248 images into four equally sized subsets and I have recruited four volunteers to view text associated with the images under four conditions. Each volunteer is a native speaker of English, and none have any connection to this or any related research. The four conditions are:

- Sent: The full first sentence of the caption.
- Rand: The words from the first sentence of the caption in random order.
- IDF: The top two words, not including proper nouns, from the first sentence of the caption, according to TF*IDF weights (Salton and Buckley, 1988; Salton, 1989).
- S-V: The two words, manually extracted, best representing the main subject and verb. If the subject is a proper noun, only the token “NAME” is provided.

Condition	Presented Text
Sent	Philippine rescuers carry a fire victim March 19 who perished in a blaze at a Manila disco.
Rand	at perished disco who Manila a a in 19 carry Philippine blaze victim a rescuers March fire
IDF	disco rescuers
S-V	subject = "rescuers", verb = "carry"

Table 6.2: The subject and verb make it clear that the category for the sample image is *Workers Responding*. Other words such as "disco" and "victim" are not helpful and can be misleading.

Table 6.2 illustrates the four conditions for the sample image shown in Figure 6.1. As is the case with many images, the subject and verb alone ("rescuers carry") are enough to confidently predict the category of the image. The top two TF*IDF words might be enough, since "rescuers" happens to be one of them, but "disco" is not helpful. If "victim" were to show up instead of "rescuers", this condition would be misleading. Viewing all the words in random order is confusing; there are mixed signals here, and unless volunteers take the time to unscramble the words and regain some syntactic clues, they are forced to guess.

A web interface has been set up that allows volunteers to predict categories of images. Each volunteer has been tested with a different condition for each of the four subsets of the data, and each subset has been presented to the four volunteers with the four different conditions. In this way, a prediction has been recorded for every image under each condition once, every volunteer has been tested under all conditions, and no volunteer has been presented with the same image twice.

Table 6.3 shows the performance of each volunteer under each condition as well as the overall performance for each condition. All volunteers are reasonably consistent. It seems that (1) more words (Sent, Rand) are better than fewer words

Volunteer	Overall Accuracy % with Specified Condition			
	Sent	Rand	IDF	S-V
#1	95.2	83.9	50.0	64.5
#2	95.2	75.8	46.8	74.2
#3	83.9	62.9	56.5	64.5
#4	90.3	75.8	61.3	83.9
Avg	91.1	74.6	53.6	71.8

Table 6.3: Subject and verb alone perform almost as well as all words in random order, and much better than the top two TF*IDF words.

(S-V, IDF), and (2) NLP helps (Sent is better than Rand and S-V is better than IDF). The NLP effect is remarkably strong and almost compensates for the other effect; i.e. Rand is only slightly better than S-V (for most volunteers). In summary, Sent $\gg$ Rand $>$ S-V $\gg$ IDF.

Condition	Average Time
Rand	68.1
Sent	34.3
IDF	22.7
S-V	20.3

Table 6.4: Volunteers spend the most time making decisions when presented all words in random order.

In addition to measuring performance, my interface also keeps track of how long each decision takes. Table 6.4 shows the average time of decisions in seconds under each of the four conditions. As can be seen, volunteers take the longest, by far, to make decisions with the Rand condition. Comparatively, with the S-V condition, they take less than one third of the time.

Examination of these results has led me to the conclusion that syntax clearly matters for this task. All volunteers perform much better when shown the full first sentence with words in their original order than when the same words are shown in random order, and the task takes approximately half the time. Therefore, any bag of words approach is likely limited by a significantly lower upper bound than one that uses NLP techniques. In particular, the main subject and verb from the sentence are important. Given only these two words, volunteers perform almost as well as when they are shown all the words in random order, and much better than when they are given the top two words according to TF*IDF weights, a very common measure of word importance in the IR literature.

6.3 Using Only Subjects and Verbs with Standard Systems

System	Overall Accuracy %	
	Sent	S-V
My systems		
BINS	56.5	53.2
Rocchio/TF*DF	57.3	53.2
Density Estimation	58.9	55.1
Rainbow systems		
Naive Bayes	55.6	54.8
Rocchio/TF*IDF	54.0	54.0
K-Nearest Neighbor	54.0	54.8
Probabilistic Indexing	59.7	54.0
Support Vector Machines	54.8	54.0
Maximum Entropy	58.1	53.2

Table 6.5: Systems perform almost as well using single word subjects and verbs as they do when provided with the entire first sentence.

I next decided to test how the standard text categorization systems I had previously tested would fair if only subjects and verbs are provided. At this point, I was still using manually extracted words. Table 6.5 compares the results using only subjects and verbs to the results using the entire first sentence of the caption (the first column of results is the same as that from Table 6.1). As can be seen, the performance is slightly worse for seven of the nine systems, slightly better for one, and the same for another. As with humans, results are almost as high using just two specifically chosen words as when all words in the sentence (not accounting for syntax) are used. According to a one-sided χ^2 test, none of the individual changes are statistically significant; however, the fact that the average performance of the nine systems decreases from 56.5% to 54.0% is marginally statistically significant (the p-value is just under 5%).

6.4 NLP Based System

With the results of the experiment with humans in mind, I set out to create a fully automatic text categorization system that takes advantage of my findings. The training phase of my system consists of examining each image in the training set and attempting to extract the single words best representing the main subject and verb from the first sentence of its caption (as described in Section 6.4.1). These extracted words are used to comprise lists of subjects and verbs that are representative of the possible categories. The testing phase of my system consists of examining the images in the test set. For each test image, the main subject and verb from the first sentence of its caption are extracted using the same method applied to the training set instances. The new subject and verb are then compared to those in the lists generated from the training set; this is accomplished by using a measure of word-to-word similarity (as described in Section 6.4.2). Finally, a score is generated for every category based on these similarities, and the category with the highest score is predicted (as described in Section 6.4.3).

6.4.1 Extracting Subjects and Verbs

Subjects and verbs are automatically extracted using a three step process. First, Church's statistical part-of-speech tagger, POS (Church, 1988), assigns a grammatical category to every word in each caption. (My own script is applied to the output of POS to correct for common errors, determined by initial experiments within the training set.) Second, the shallow parser CASS (Abney, 1997) parses each tagged caption. Third, a final script operates on the output of CASS, extracting the heads of the appropriate noun phrase and verb phrase to obtain the single words assumed to best represent the subject and verb of the sentence. (If CASS considers the head of the noun phrase to be a name, the token "NAME" is used instead). WordNet is used to convert each extracted subject and verb to its morphological base-word. This process is used to extract subjects and verbs for the training set documents (in order to comprise lists of subjects and verbs that are indicative of each category) and also for the test set documents (in order to predict a category for each image). When applied to the training set, this process leads to an accuracy of 92.7% for subjects and 86.3% for verbs, according to the manually extracted words. When applied to the test set, this process leads to an accuracy of 83.9% for subjects and 80.6% for verbs, according to the manually extracted words. The reason that the accuracy is higher for the training set is that I allowed myself to examine the training documents when implementing the script used to extract the heads of the appropriate nouns and verb phrases. Clearly, the rules that I came up with worked reasonably well for the test set as well.

6.4.2 Word Similarity

Since this system is limiting itself to only two words per document (those best representing the main subject and verb from the first sentence of each image caption), and the corpus is small relative to those used for most text categorization tasks, it is likely that words may occur that have never been seen before. For this reason, direct comparison of words to each other, as is done by most bag of words approaches, is not appropriate. Some method of word-to-word similarity is necessary to handle new words, or it would not be possible to make a prediction for many test documents. In Section 6.7, I describe some of the research involving techniques for computing word-to-word similarity, and also why these techniques are not appropriate for the task described in this chapter. Instead, I use the approach explained in this section, which is somewhat novel to the best of my knowledge.

In order to compare subjects and verbs extracted from test captions to those from the training set, the system examines a large "extended" corpus consisting of thousands of news articles and embedded captions that are contained in my corpus, including unlabeled documents (as described in Section 3.1.1). Using the same method of extraction as discussed in Section 6.4.1, the single words best representing the subjects and verbs are extracted from every sentence of every article and caption in the extended corpus. When dealing with text categorization, the creation of the corpus is generally one of the most time consuming tasks, since documents usually need to be manually labeled for the training set; however, for the purposes of word similarity as I am calculating it, the extended corpus does not need to be labeled, and it is thus easily obtainable.

Based on these extracted subject/verb pairs, the similarity between two subjects is defined to be the percentage of verbs they share in common, and the similarity between two verbs is defined to be the percentage of subjects they share in common. The idea is that two subjects should be considered similar if they often partake in similar actions, and that two verbs should be considered similar if they represent actions that are often executed by similar entities. This is not necessarily a good measure of word similarity for other tasks, but I thought it might work well for this domain. For example, let's say that the word "fireman" never appears in the training set of the corpus, but words such as "policeman" and "volunteer" do, in captions from images belonging to the category *Workers Responding*; these subjects likely share a higher percentage of verbs in common than most randomly selected pairs of subjects, and would therefore have a relatively high similarity. In addition, for the current domain, they are representative of the same category (*Workers Responding*). By my definitions, the similarity between any subject or verb and itself comes out to be one, and the similarity between any two non-identical words is generally much less.

I also defined the similarity between a subject and a verb to be twice the number of times they appear together divided by the total number of times each appears. Therefore, if the subject/verb pair always appears together, the similarity between the two words is one, and otherwise it is less. The idea is that subjects that are likely to perform actions seen as representative of a category should, in and of themselves, be considered representative of the category. The same is true for verbs that represent actions that are likely to be performed by subjects that are representative of a category. For example, let's say that the word "fireman" never appears in the training set of the corpus, but verbs such as "help" and "rescue" do, in captions from images belonging to the category *Workers Responding*. Since a "fireman" is more likely to "help" and "rescue" than perform other activities, it contributes more to the *Workers Responding* category than to others.

6.4.3 Choosing a Category

Once the single word subject and verb from the first sentence of an image's caption are extracted using the process described in Section 6.4.1, they must be used to predict a category for the image. First, the extracted subject and verb are both compared to every subject and verb extracted from the training set documents (also obtained through the process described in Section 6.4.1). Each comparison involves computing a similarity between two words using the measure of word-to-word similarity explained in Section 6.4.2. For each possibly category, all similarities involving words extracted from training set documents belonging to the category are summed together; this sum represents a score for the category, and the category with the highest score is then predicted. (There is one additional step if the extracted subject is a "NAME" token. This is explained later in the section.)

More formally, let C be the set of categories, and for some specific category c, let S_c and V_c be the set of subjects and verbs extracted from training instances of c, respectively. For a particular test image d, let s_d and v_d be the single word subject and verb extracted, respectively. For any two words w_1 and w_2, regardless of whether they are subjects or verbs, let Sim_{w_1,w_2} be the similarity between the two words as defined in the Section 6.4.2 (any similarity involving a "NAME" token is defined to be 0). Let $T(c|d)$ be the total score for a category c given a test document d. Then:

$$T(c|d) = \left(\begin{array}{c} \sum_{s_c \in S_c} [Sim_{s_d,s_c} + Sim_{v_d,s_c}] \\ + \sum_{v_c \in V_c} [Sim_{s_d,v_c} + Sim_{v_d,v_c}] \end{array} \right) \tag{6.1}$$

For a document d that does not have a "NAME" token extracted as the subject, the chosen category is simply:

$$\underset{c \in C}{argmax}[T(c|d)] \tag{6.2}$$

In order to take "NAME" tokens into account when they are extracted (this occurs in 16 of the 124 test cases), I decided to multiply the score for each category by the *a-priori* probability of the category, based on the training set, given that a "NAME" token is extracted. For example, in the training set, 56.0% of the "NAME" tokens come from the *Affected People* category, whereas only 33.9% of the training images belong to this category overall, so the final score for the *Affect People* category is multiplied by 0.56 if a "NAME" token is extracted to account for the new skew. More formally, let $P(NAME|c)$ be the estimated probability of a "NAME" token given a category, based on the training set. Then the category chosen for a document d that has a "NAME" token extracted is:

$$\underset{c \in C}{argmax}[T(c|d) \times P(NAME|c)] \tag{6.3}$$

6.5 Results and Evaluation of NLP Based System

System	Overall Accuracy %	
	Sent	S-V
My systems		
NLP Based System	—	65.3
BINS	56.5	53.2
Rocchio/TF*DF	57.3	54.0
Density Estimation	58.9	55.1
Rainbow systems		
Naive Bayes	55.6	54.8
Rocchio/TF*IDF	54.0	54.0
K-Nearest Neighbor	54.0	54.8
Probabilistic Indexing	59.7	54.0
Support Vector Machines	54.8	54.0
Maximum Entropy	58.1	53.2

Table 6.6: The NLP based system outperforms nine standard systems by a considerable margin.

The first line of Table 6.6, which is otherwise the same as Table 6.5, shows the performance of my NLP based system described in the previous section. As can be seen, the system's overall accuracy of 65.3% is at least 10% higher than the nine standard bag of words systems achieve when using only subjects and verbs (manually extracted for the standard systems), and it is at least 5% higher than the nine standard systems achieve when they are provided the entire first sentence of each caption. According to a one-sided χ^2 test, the gain over the standard systems using only subjects and verbs is statistically very significant (the performance of the closest competitor leads to a p-value of about 1%), whereas only the top three competitors relying on full sentences fall within a 95% confidence interval. The use of deeper linguistic processing has thus led to a nice performance gain for this particular task. Looking back at Section 6.2, we see that humans given only the subject and verb of each sentence achieve, on average, a 71.8% accuracy. This should be considered a reasonable upper bound for the accuracy that a system such as mine might achieve.

6.6 Experiments with the Politics Image Data Set

In order to test the generality of the approach discussed in this chapter, I have applied all of the systems tested on the *Disaster* image data set to the *Politics* image data set, first described in in Section 3.1.2.5. This data set consists of images embedded in news documents that belong to the Events category *Politics*. As a reminder, this data set consists of 299 images; 86 (28.8%) are classified as *Meeting*, 64 (21.4%) are classified as *Announcement*, 88 (29.4%) are classified as *Politician Photographed*, 40 (13.4%) are classified as *Civilians*, 14 (4.7%) are classified as *Military*, and 7 (2.3%) are classified as *Other*. The data set has been randomly divided into a training set containing 149 images and a test set containing 150 images.

Based on some initial experiments within the training set, two minor changes have been made to the NLP based system. First, the script I apply to the output of POS to correct common errors (as mentioned in Section 6.4.1) has been adjusted. Second, the final phase of the system that recalculates probabilities when a "NAME" token is extracted as the subject has been left out. Therefore, only verbs are being used for these cases. Whereas this only occurs in about 13% of the test instances for the *Disaster* image data set, it occurs in about 60% of the test instances for the *Politics* image data set. Although this actually means it is important to account for "NAME" tokens in some manner, the particular method used for the *Disaster* image data set is not appropriate. This is discussed further later in the section.

System	Overall Accuracy %	
	Sent	S-V
My systems		
NLP Based System	—	54.7
BINS	53.3	46.7
Rocchio/TF*DF	56.7	46.7
Density Estimation	64.7	49.3
Rainbow systems		
Naive Bayes	53.3	53.3
Rocchio/TF*IDF	54.7	52.7
K-Nearest Neighbor	36.0	36.7
Probabilistic Indexing	64.0	56.0
Support Vector Machines	53.3	52.7
Maximum Entropy	56.7	51.3

Table 6.7: For the task involving the categorization of images in the *Politics* image data set, the NLP based system outperforms eight out of nine standard systems when the standard systems are provided only subject and verb, and it is in the middle of the pack when the standard systems are provided full sentences.

Table 6.7 shows the results of all systems tested for the *Politics* image data set. This time, the NLP based system outperforms eight out of nine standard bag of words systems when the standard systems are provided only the main subject and verb. When the standard systems are provided the entire first sentence, the NLP based system beats four, loses to four, and ties one, so it is right in the middle of the pack. It is interesting to note that the range of performance of standard systems for the *Politics* image data set (kNN's 36.0% to density estimation's 64.7%) is much larger than the range for the *Disaster* image data set (kNN's and Rocchio/TF*IDF's

54.0% to Probabilistic Indexing's 59.7%). Taking these results as a whole, it seems as though the method of word-to-word similarity that the NLP based system is using is helpful, since this system is beating the standards when they use the same words (the main subject and verb). However, the benefit of linguistic processing is not overcompensating for the drawback of less words; the two are approximately cancelling each other out in this case. This is not to say that linguistic processing can not overcompensate for these categories if it is done just right; rather, the specific methodology set up with the *Disaster* image data set in mind is not as appropriate for the *Politics* image data set.

The probable reason that the NLP based system is not as successful for the *Politics* image data set as it is for the *Disaster* image data set is that the main subjects of captions for the *Politics* images are much more likely to be proper nouns. As mentioned earlier in the section, only 13% of the test instances in the *Disaster* image data set have "NAME" tokens extracted, whereas 60% of test instances in the *Politics* image data set have name tokens extracted. In addition, inspection reveals that the proper name subjects associated with the *Politics* images seem to be more complicated than those associated with the *Disaster* images, often consisting of a list of multiple names. For the *Disaster* image data set, the system takes "NAME" tokens into account by multiplying category scores by the *a-priori* probabilities of the categories, based on the training set, given that a "NAME" token is extracted; for that data set, it turns out that this leads to a small improvement in overall accuracy (less than 1%). However, experiments within the training set have indicated that this is not helpful for the *Politics* image data set. (Out of curiosity, I did eventually run an experiment including this rule for this data set, and the accuracy degrades slightly, from 54.7% to 52.7%.) So, for 60% of the *Politics* image documents, only one word (the main verb) is being used to predict the correct category. Given this fact, it is impressive that the system performs as well as half of the standard systems when they are provided the entire first sentence of each caption. In order to have the NLP based system achieve a performance superior to that of all the standard bag of words systems for the *Politics* image data set, it is probably necessary to find a better way of handling proper name subjects for these categories. I leave that for future work.

6.7 Research Related to NLP and Text Categorization

There has long been a lot of interest in combining NLP and IR. Some of the recent work by various researchers in this area is summarized in (Strzalkowski, Lin, and Perez-Carballo, 1998) and (Strzalkowski, 1999), and as can be seen from these sources, the results have been mixed, at best. Recently, there has been some success using NLP to aid in the retrieval of images. Smeaton and Quigley (1996) show some improvement using WordNet (Fellbaum, 1998) to compute noun-to-noun similarities that are then used to compare queries with captions. Elworthy (2000) shows improvement using an NLP technique he calls "phrase matching" which first converts queries and captions to "dependency structures". In both of these cases, the researchers manually construct captions for their images, and in the case of Smeaton and Quigley, they manually disambiguate all words.

Working on domain-specific text categorization tasks involving full-length news articles, Riloff has created the system AutoSlog-TS (Riloff, 1996; Riloff and Lorenzen, 1999) which relies on NLP techniques to automatically create dictionaries of "augmented relevancy signatures" that can then be used to improve results for binary text categorization tasks. She has found that her system, which labels a document in a category if any augmented relevancy signature associated with the category is found in the document, performs about as well with automatically constructed dictionaries as it does with hand constructed dictionaries and much better than when no dictionary is used at all. Riloff does not compare her system to other standard text categorization techniques.

With IR tasks such as query expansion and word sense disambiguation in mind, there have been previous attempts at measuring word-to-word similarity. The research discussed in (Sussna, 1993), (Resnik, 1999), and (Richardson, Smeaton, and Murphy, 1994) concerns using WordNet link structure to determine semantic similarity between nouns. My task additionally requires the system to compute similarities between verbs with each other and verbs with nouns, so the techniques discussed in these papers do not apply. My approach is simpler, and not necessarily appropriate for general tasks, but it serves my intended purpose well and leads to positive results in my experiments.

Other commonly used metrics to measure word-to-word similarity for use with NLP applications include the Jaccard Coefficient and the Dice Coefficient (Radecki, 1982; van Rijsbergen, 1979; Smadja, McKeown, and Hatzivassiloglou, 1996). These measures are related to the ratio of the frequency with which two words appear together (i.e. near each other) in text to the frequencies of the two words independently. While simple and general, they do not apply well to the specific task and domain discussed in this chapter. For example, "rescuers" and "victim" might often appear together in text, as they do in the caption of the sample image in Figure 6.1, but for the current categorization task, as subjects they would be indicative of two different categories (*Workers Responding* versus *Affected People*), and they should not be considered similar. On the other hand, words such as "firefighter" and "fireman" may hardly ever appear together, since an author will likely use one or the other consistently, but for this task, they should be considered very similar. My method of measuring word-to-word similarity takes these problems into account.

The work discussed in (Hatzivassiloglou and McKeown, 1993) and (Hatzivassiloglou, 1998) concerns the identification of groups of related adjectives. One criteria that the authors use to indicate whether or not two adjectives are related is the tendency of the two words to modify the same nouns. In other words, adjective/noun pairs are extracted from a corpus (such that the adjective modifies the noun), and two adjectives are considered related if they tend to co-occur with the same nouns (determined using an estimate of Kendall's τ based on the extracted word pairs). This is very similar to the method I describe in Section 6.4.2 for measuring the similarity between two subjects or two verbs. In my work, I am extracting subject/verb pairs as opposed to adjective/noun pairs, and the metrics I am using to evaluate similarity based on the obtained word pairs are simpler.

6.8 Concluding Discussion of NLP and Text Categorization

I have shown that NLP is important for a particular text categorization task involving categories that distinguish objects of focus in the foreground of an image from objects that may be present in the background of an image. I believe that the importance of NLP depends on both the task and domain. NLP becomes helpful when we are dealing with tasks that rely on focus, perspective, point of view, etc. Admittedly, most of the standard IR test collections are not like this, and bag of words approaches work well for them. However, I believe that tasks such as the one described in this chapter, which arose naturally in the course of my research, will continue to appear, and when they do, approaches similar to this will be useful. I further believe that this is more likely to happen when dealing with images compared to full-length text documents. A recurring theme of this dissertation is that categories applied to images are often quite different than those applied to standard text categorization documents. When dealing with images, it seems natural that one might be interested in the object of focus and the action taking place; words referring to background information are thus not as important, and syntax therefore matters.

The *Disaster* image categories discussed in this chapter are not nominal categories determined by the presence or absence of any specific object in an image. These categories deal with predicate argument relationships that can only be determined using linguistic analysis. Looking back, once again, at Figure 6.1, we see that the subject and verb of the first sentence of the caption refer to the object of focus in the image and the action taking place. The phrase “rescuers carry” is a clear indication of the *Workers Responding* category, whereas other words that might have high IDF weights, such as “disco” and “victims”, would not be helpful and may even be misleading to any system using a bag of words approach. I have verified the importance of NLP for the task by presenting evidence from experiments with human subjects, and I have described a new NLP based system that considerably outperforms nine standard systems. This is a positive result that shows promise for combining NLP and IR in the future, at least for certain tasks.

Chapter 7

High-Precision/Low-Recall Rules

I have shown in Chapter 6 that consideration of syntax is necessary for optimal performance for the *Disaster* image data set. The results of the experiments with human volunteers discussed in Section 6.2 make this clear. It is likely that the same is true for the *Politics* image data set, although the analogous experiment with humans has not been carried out. In Chapter 6, I have also described an NLP based system that considerably outperforms all standard systems tested on the *Disaster* image data set, and it is competitive when tested on the *Politics* image data set. However, it is likely that this system is still not optimal, as the performance for the *Disaster* image data set (overall accuracy is about 65%) is still nowhere near the upper bound achieved by humans when they are shown the entire first sentence of the caption (overall accuracy is over 90%), and it is still almost 10% under the performance of humans when they are shown all the words of the first sentence in random order (overall accuracy is about 75%).

The work discussed in this chapter originated with the idea that while subjects and verbs are clearly very important (demonstrated both by the results of experiments with humans, and also by the results of systems), it is possible that other words, while less important on average, might sometimes offer very good clues.

It is possible that a system that generally uses the main subject and verb from each caption to make its predictions may, at times, be better off changing the prediction based on certain other words. For example, looking ahead to figure 7.1, the image shown is a member of the *Affected People* category of the *Disaster* image data set. The main subject and verb of the first sentence of the image's caption are a proper noun, which is slightly indicative of the *Affected People* category, and a form of the verb "carry", which is twice as indicative of the *Workers Responding* category as it is of the *Affected People* category. The NLP based system described in Chapter 6 gets this example wrong. However, the rest of the first sentence makes it clear to a human that the associated image belongs to the category *Affected People* category, and it turns out that two words in this sentence - "family" and, surprisingly, "his" - are statistically extremely indicative of this category as well.

In this chapter, I describe a semi-automated procedure by which such words can be discovered. A list of such words is generated, and each word in the list is associated with a corresponding category. In other words, the list is actually a mapping of each selected word to a category that is heavily indicated by the word. If any such word is found in a document, the category indicated by the word is predicted; otherwise, the system falls back to its normal technique.

The idea here is that while most documents do not contain any word from the list (therefore, use of the list alone would achieve low-recall for any given category), when such a word is contained in a document, there is a very high probability that the indicated category associated with it is correct (therefore, the list alone would achieve high-precision for all categories based on the documents to which it applies). As long as the accuracy using the list is greater than that of the fall back technique *when the list applies*, the overall performance of the system is improved. Although this idea was originally intended specifically to improve the NLP based system described in Chapter 6, it turns out that adding such a first-pass technique to other systems also almost always seems to improve performance for the *Disaster* image data set and the *Politics* image data set. These two data sets are discussed in detail in Sections 3.1.2.4 and 3.1.2.5.

7.1 Choosing Indicative Words

In order to semi-automatically discover words, based on training data, that may be highly indicative of a particular category, two things are important. One is that the word appears mainly in the category of interest; i.e. there must be a statistical bias towards a particular category given that the word appears in a document. The other is that the word must appear enough times to make this trend noteworthy. If a word only appears once in the training data, it doesn't matter that the word occurs only in one category.

The method I use to discover such words is similar to the method used by Riloff (1996; 1999) to discover augmented relevancy signatures (I provide some more detail about this in Section 7.5). I have written a script that examines all training data and accepts two cutoffs from the user. One cutoff, which I will call X, represents the minimum number of documents in which a word must occur, and the other cutoff, which I will call P, represents the minimum proportion of occurrences of the word that must appear in a single category. In other words, if a single word appears x times with $p \times x$ of those occurrences in a single category, where $x >= X$ and $p >= P$, then this word qualifies as being appropriately indicative of the category. The process as a whole is only semi-automated; I view the outputs resulting from various values of X and P, choosing the words that seem appropriate and pruning those that seem to appear mainly in one category by coincidence. For these experiments, a "word" is defined to be anything recognized as such by Church's statistical part-of-speech tagger, POS (Church, 1988). All words other than proper nouns are converted to lower case.

Word	Indicated Category	Total Count (x)	Proportion (p)
her	Affected People	7	1.0
his	Affected People	7	0.86
family	Affected People	6	0.83
relatives	Affected People	6	1.0
rescue	Workers Responding	15	1.0
search	Workers Responding	9	1.0
similar	Other	2	1.0
soldiers	Workers Responding	6	1.0
workers	Workers Responding	12	1.0

Table 7.1: These words are highly indicative of the specified categories within the *Disaster* image data set. Some, such as “her” and “his”, are not intuitively obvious.

Table 7.1 shows the words that I selected for the *Disaster* image data set. The second column shows the category that seems to be indicated by the word in the first column. The third column shows x, which is the total number of documents that contain the word in the training set, and the fourth column shows p, the proportion of these x documents that actually belong to the indicated category. (As always with this data set, only the first sentences of captions are used, as described in Section 3.1.3.) Note that there are no words in the list that indicate the *Wreckage* category, and there is only one word that indicates the *Other* category (this word, “similar”, is discussed more below). Most of the words in the list seem intuitive, but there are some exceptions. The words “her” and “his”, for example, do not sound like they should be indicative of any particular category, but the statistical evidence is strong. Each word appears in seven documents in the training set (without any overlap), and 13 of these 14 documents belong to the *Affected People* category.

Table 7.2 show all of the sentences in the training set of the *Disaster* image data set that contain either of the words “his” or “her”. No sentence contains both words. 13 of the 14 associated documents belong to the *Affected People* category,

category	sentence
Affected People	Dan Crull walks to *his* boat after giving *his* wife a ride to their vehicle (L), parked on high ground near their home in the flooded village of Rome, February 28.
Affected People	President Clinton hugs Judy Sligh March 4, whose restaurant was destroyed by a tornado March 1, during *his* visit to Arkadelphia.
Affected People	A resident navigates through *his* flooded neighborhood in *his* boat April 10 as the Red River continues to rise in the upper Midwest.
Affected People	A father weeps, May 11, next to the body of *his* child killed during a massive earthquake that struck northeast Iran May 9.
Affected People	An elderly man abandons the ruins of *his* house July 10 after planting a national flag on the remains.
Other	Haitian President Rene Preval (center) observes body recovery operations from the sunken ferry at Montrouis, September 9, with members of *his* government, Health Minister Rudolphe Malebranche (L) and Prime Minister Rosny Smarth (R) Preval declared three days of national mourning for the tragedy, which cost the lives of possibly more than 200 people.
Affected People	Higinio Guereca carries family photos he retrieved from *his* mobile home which was destroyed as a tornado moved through the Central Florida community, early December 27.
Affected People	Seven year-old Jessica Dubroff stands with *her* father, Lloyd, prior to taking off from Half Moon Bay Airport near San Francisco April 10.
Affected People	Judy Lyncher Teller shows pictures of *her* nieces Shannon, 10, and Katie, 8, who were killed in the ill-fated TWA flight 800, to reporters outside a hotel at John Kennedy airport July 19.
Affected People	Jean Scupp sits in solitude in *her* beachfront home which had been pushed several feet away from its footing and probably destroyed by hurricane Fran September 7.
Affected People	Buffalo resident Noelle Silbak shovels out *her* car on Germain Street January 12.
Affected People	A young girl walks through *her* flooded living room October 12 after hurricane Pauline devastated Mexico's most famous tourist resort.
Affected People	A Somali refugee walks with *her* two children, November 27, through a flooded refugee camp located in north eastern Kenya.
Affected People	Lisa Nichols (L) and *her* sister-in-law Nancy Franklin (R) embrace April 11 amid the ruins of Nichols' grandmothers' Rock Creek, Alabama residence.

Table 7.2: The sentences from the *Disaster* image training set containing either "his" or "her" present strong evidence that these words are highly indicative of the *Affected People* category.

and in 12 out of these 13 cases, the word "his" or "her" refers directly to an affected person who is likely the focus of the image. In the one remaining *Affected People* image, the word "his" refers to President Clinton, and in the case of the document belonging to the *Other* category, the word "his" also refers to a politician visiting the scene. The fact that so many instances of these words appear in *Affected People* training documents, that they refer to an actual affected person in most cases, and that these words rarely occurs in training documents of other categories presents strong evidence that the words should remain in the list. This is not something that would have been predicted by someone who was comprising a similar list manually.

This final list is very similar to the one that is found automatically by the script I have written when it is given the parameters $X = 6$ and $P = 0.83$ (just under 5/6). The only difference is that I have added the word "similar" as indicative of the *Other* category. This word appears in the list when $X = 2$ and $P = 1$, and I had previously noticed that the word "similar" is indicative of the *Other* category in an interesting way.

category	sentence
Other	(UNDATED FILE PHOTO) An Indonesian Airbus A-300 passenger jet, *similar* to the one shown in this file photo, crashed in northern Sumatra, killing all 234 passengers and crew according to state television, September 26.
Other	(UNDATED FILE PHOTO) An Air Force T-38 Talon jet training plane, *similar* to the Northrop Grumman T-38's shown, crashed after colliding in mid-air with an F-16 Falcon fighter jet over Edwards Air Force Base in California October 22, an Air Force spokesman said.

Table 7.3: The sentences from the *Disaster* image training set containing the word "similar" present evidence that this word is indicative of the *Other* category.

Table 7.3 shows all of the sentences in training set of the *Disaster* image data set that contain the word "similar". Both of the corresponding images are classified as belonging to the *Other* category, and in each case, the word "similar" describes a relationship of the object of focus of the image to some other object involved with the disaster that has taken place. The word "similar" does not occur in any document of any other category. Perhaps some might claim that just two instances is not enough to justify keeping the word in the list (I have not included any other word that occurs with such a low frequency), but this is an interesting case involving the toughest category, so I have kept it in the list. In any case, it turns out that the word "similar" only occurs in one test document, so keeping this word in the list does not have a significant effect on results (in this one case, "similar" is indeed used in a manner like the two shown in the table, and the *Other* category is the correct prediction).

Word	Indicated Category	Total Count (x)	Proportion (p)
hands	Meeting	10	0.90
journalists	Announcement	4	1.0
local	Civilians	4	1.0
media	Announcement	3	1.0
presidential	Politician Photographed	9	0.78
press	Announcement	7	0.71
reporters	Announcement	8	0.88
meeting	Meeting	15	0.73
session	Meeting	6	0.83
victory	Politician Photographed	6	0.83
waves	Politician Photographed	4	1.0
wife	Politician Photographed	6	1.0

Table 7.4: These words are highly indicative of the specified categories within the *Politics* image data set. Some, such as "hands" and "wife", are not intuitively obvious.

Table 7.4 shows the words that I selected for the *Politics* image data set. The second column shows the category that seems to be indicated by the word in the first column. The third column shows x, which is the total number of documents that contain the word in the training set, and the fourth column shows p, the proportion of these x documents that actually belong to the indicated category. (Once again, only the first sentences of captions are used, as described in Section 3.1.3.) Note that there are no words in the list that indicate the *Military* category or the *Other* category. As with the words chosen for the *Disaster* image data set, most of the words in the list seem intuitive, but there are some exceptions, such as “hands” and “wife”. It turns out that “hands” almost always occurs in the context of *shaking hands*, and this is something that often happens at the start or end of a meeting; apparently, journalists are likely to take a picture of it. The word “wife” almost always occurs in the context of a politician being photographed with his wife.

This final list is somewhat similar to the one that is found automatically by the script I written when it is given the parameters $X = 4$ and $P = 0.83$ (just under 5/6). This means that words that occur four or five times must appear only in one category to be included, words that appear six through 11 times may appear once in another category, words that appear 12 through 17 times may appear twice in another category, etc. I have added four words that do not meet this criteria. One of these words is “media”, which only occurs three times in the training set but the intuition behind it is clear. The other three are “presidential”, “press”, and “meeting”, which all occur in a category other than the indicated category one or two more times than allowed when $P = 0.83$, but these words show up in the list if I lower the value of P, and, again, the intuition behind these words is strong. I have also pruned a few words that show up automatically when $X = 4$ and $P = 0.83$, such as “president” (which occurs six times, with five of the occurrences in the *Politician Photographed* category) and “before” (which occurs six times, with five of the occurrences in the *Meeting* category) for reasons explained further below.

Table 7.5 shows all of the sentences in training set of the *Politics* image data set that contain the word “president”. Although five of the six corresponding

category	sentence
Politician Photographed	Monsignor Laurent Monsengwo (C), the Catholic Archbishop who was elected to be leader of parliament and vice *president* arrives in Kinshasa May 12, after crossing over the River Zaire from Brazzaville.
Politician Photographed	Montenegro's outgoing *president* Momir Bulatovic and his wife Nada cast their votes October 19 in the presidential run-off seen as crucial for the tiny Balkan republic and its ties with neighbor and Yugoslav federation partner Serbia.
Meeting	Cheryl Carolus (L), acting general secretary of the ruling African National Congress (ANC), ANC chairman Jacob Zuma (C) and Thabo Mbeki, ANC deputy *president*, gaze at delegates at the party's 50th congress at Northwest University in Mmabato, northwest of Johannesburg, South Africa, December 16.
Politician Photographed	South African President Nelson Mandela (R) congratulates his deputy Thabo Mbeki on his election December 17 to succeed him as *president* of the ruling African National Congress.
Politician Photographed	Muhammad Rafiq Tarar, Pakistani premier Nawaz Sharif's party nominee for *president*, casts his vote in the presidential elections December 31.
Politician Photographed	Former Indonesian *president* Suharto (C) is surrounded by security guards as he is shown the way to the High Prosecutor's Office December 9.

Table 7.5: The sentences from the *Politics* image training set containing the word "president" do not present convincing evidence that this word is indicative of the *Politician Photographed* category.

images are members of the *Politician Photographed* category, the word is not used in such a way that it, in itself, is indicative of the category, and it is used in roughly the same manner in the example corresponding to an image in the *Meeting* category. Therefore, this word does not seem to be highly indicative of the *Politician Photographed* category, and instead seems to have only appeared mostly in this category by coincidence. The other words I have pruned from the list that occurs when $X = 4$ and $P = 0.83$ have been pruned for similar reasons.

7.2 Using Indicative Words to Improve Categorization

The lists of words selected as just described in Section 7.1 can be used as follows. Given any document, first the list associated with the current data set is scanned to see if any word from the list is contained in the document. If so, the category associated with that word is predicted. Otherwise, some other text categorization system is used to predict the category of the document. (If more than one word from the list are contained in the document, and the words indicate contradicting categories, my system uses only the first such word found. Another option would be to ignore the words from the list and fall back to the other categorization system when this happens. In any case, it is extremely rare, and, in fact, I do not believe that this ever happens with either of the two data sets discussed in this chapter.)

Each word in the list associated with a data set can be thought of as constituting a single rule. The rule for a word w associated with a category c would simply be "if w exists in the document, predict category c". These rules are similar to the rules that comprise decision lists[1], which is discussed in Section 7.5. Usually, a decision list ends with a default category that is chosen when no other rule applies (the category that is statistically most likely, according to the training data, in this case); however, that is almost certainly not the best thing to do in cases such as that which I am dealing with here, when most documents are not handled by the rules. So instead, I am falling back to some other system, thereby combining an approach reminiscent to a decision list approach with that of the fall back system.

Not all documents should be expected to contain a word from the list. In fact, it is usually the case that no such word is found; therefore, the rules associated with the list are "low-recall" rules. In and of themselves, they are not sufficient for

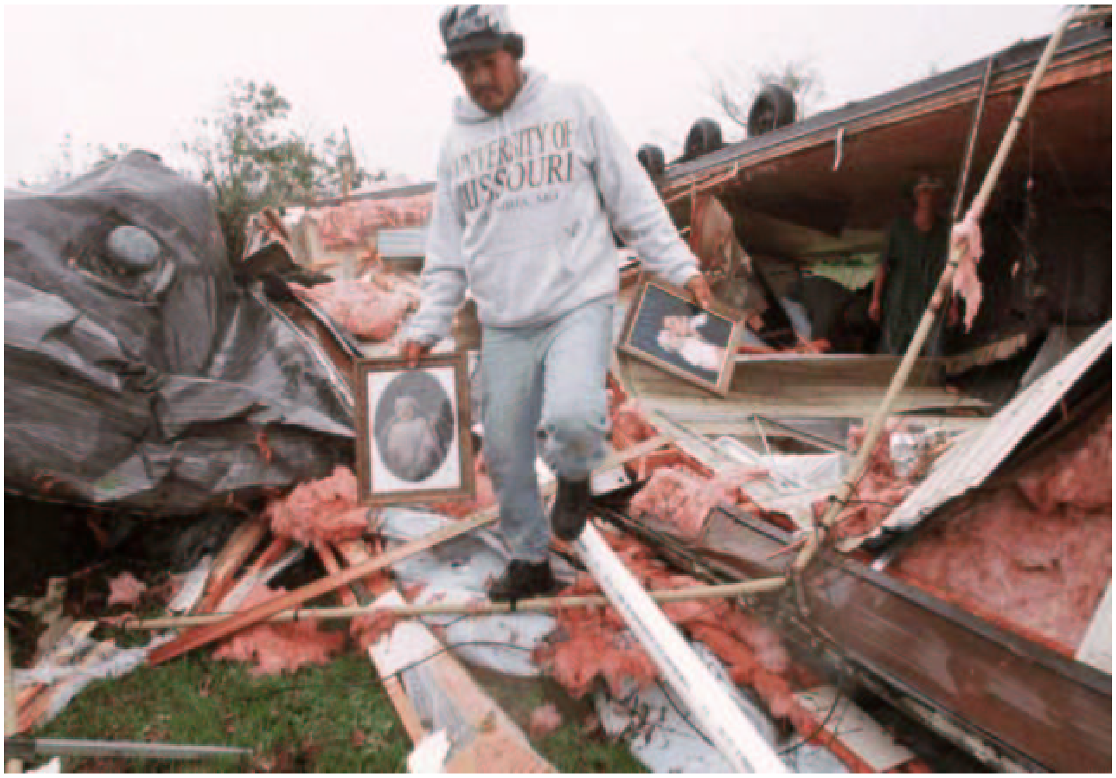
[1]Or decision trees, but since my system examines the words (rules) in a linear fashion, and uses only the first that applies, a decision list makes for a better analogy.

discovering most members of most categories. When no rules applies, the prediction is made by some other system. The accuracy for these documents (those that do not contain any word from the list) is exactly the same as the accuracy for the backup system used. For the documents that do contain a word from the list, the rule associated with that word, and that rule alone, determines the prediction of the category. This leads to an overall improvement in accuracy over the backup system if (and only if) these rules are better predictors than the backup system only considering the cases for which a rule applies. If the selection of words (and therefore rules) is done appropriately, each rule should typically be correct when it applies; therefore the rules associated with the list are "high-precision" rules. When a rule places a document into a category, it is likely correct.

7.3 Experiments with High-Precision/Low-Recall Rules

Originally, I conceived of this two pass approach (first using the high-precision/low-recall rules and backing off to some other system when no rule applies) specifically to improve the NLP based system described in Chapter 6. The NLP based system only uses two words in each document (the main subject and the main verb) to make its prediction. While these two words have been shown to be particularly important for the *Disaster* image data set (and they are likely important for the *Politics* image data set), it is still possible, and even likely, that other words may offer good clues in certain cases.

Figure 7.1 shows an image from the *Affected People* category of the *Disaster* image data set and the first sentence of its caption. Although wreckage is also seen in the image, the focus is on the victim whose home was destroyed. In this case, the main subject of the sentence is a proper noun and the main verb is "carries"



Higinio Guereca carries family photos he retrieved from his mobile home which was destroyed as a tornado moved through the Central Florida community, early December 27.

Figure 7.1: The subject and verb here do not offer good clues, but the words "family" and "his" have corresponding rules which correctly predict the *Affected People* category.

(which WordNet maps to the baseword "carry"). Remember from Section 6.4.3 that proper names as subjects do turn out to be slightly indicative of the *Affected People* category; however, the verb "carry" turns out to be twice as indicative of the *Workers Responding* category as it is of the *Affected People* category (which makes sense, since rescue workers sometimes carry victims from the scene of a disaster). The NLP system gets this example wrong. However, the words "family" and "his" both appear in the sentence, and either of the two rules associated with these words correctly maps this image to the *Affected People* category.

Although this two-pass technique was originally intended to improve the NLP based system, there was no reason not to test the approach using other fall back systems as well. Therefore, I have tested adding these high-precision/low-recall rules as a first-pass approach to each of the Rainbow systems (results of these systems for the *Disaster* image data set and the *Politics* image data set are first presented in Sections 3.2.3 and 3.2.4), to my BINS system (using bins combined with Naive Bayes as described in Section 5.7), and also to my NLP based system. The results, described in the next section, surprised me.

7.4 Results and Evaluation of the Two-Pass Approach

Table 7.6 shows the results the eight systems used for this experiment when applied to the *Disaster* image data set. The first column of results shows the overall accuracy of each system on its own (i.e. the system is used for every test document), while the second column of results shows the overall accuracy of each system combined with the high-precision/low-recall rules (i.e. the system is a fall back used when no rule applies, meaning that no highly indicative word from the appropriate list is found in the document). As you can see, the addition of the first-pass

System	Overall Accuracy %		
	Without Rules	With Rules	Change
My systems			
NLP Based System	65.3	67.7	+2.4
BINS	56.5	63.7	+7.2
Rainbow systems			
Naive Bayes	55.6	59.7	+4.1
Rocchio/TF*IDF	54.0	60.5	+6.5
K-Nearest Neighbor	54.0	59.7	+5.7
Probabilistic Indexing	59.7	66.1	+6.4
Support Vector Machines	54.8	59.7	+4.9
Maximum Entropy	58.1	61.3	+3.2

Table 7.6: The addition of a first-pass using high-precision/low-recall rules improves the performance of all of the eight systems tested for the *Disaster* image data set.

improves the performance of every system tested for the *Disaster* image data set. The system that improves the least (2.4%) is the NLP based system, perhaps because it does the best to begin with (and still does the best when combined with the new rules). The system that improves the most (7.2%) is my BINS system. Notice that of all the bag of words approaches, this is the only one that does not compute weights of for individual words, and perhaps this is why there is room for improvement when there are single words that are highly indicative of a category. The average change in performance over all systems is an improvement in overall accuracy of 5.1%.

As described in Section 3.1.2.4, the test set for the *Disaster* image data set contains 124 images. It turns out that the rules corresponding to the words listed in Table 7.1 apply to 39 (31.5%) of these 124 images. In other words, 39 of the 124 test images have one or more of the words from the list appear in the first sentence of the caption. In 33 (84.6%) of these 39 cases, the category associated with the word from the list is the correct prediction. The fact that every system in Table 7.6

improves when the high-precision/low-recall rules are used as a first pass means that none of these systems perform as well as the rules for these 39 documents.

According to a one-sided χ^2 test, the p-values for the individual changes in performance across the entire test set range from 1.2% to 31.7%. Most of the individual changes are not statistically significant at the 95% confidence level, but all are still unlikely to occur by chance. Testing the average improvement for all systems (over all decisions being made) leads to a p-value of 0.07%, which is extremely statistically significant. (Really, all of these p-values should probably be computed only based on the examples to which the rules apply, in which case all of the p-values would be lower, and the significance levels therefore higher, since the same number of examples are changing from wrong to right based on a much smaller set.)

System	Overall Accuracy %		
	Without Rules	With Rules	Change
My systems			
NLP Based System	54.7	62.7	+8.0
BINS	53.3	59.3	+6.0
Rainbow systems			
Naive Bayes	53.3	59.3	+6.0
Rocchio/TF*IDF	54.7	58.7	+4.0
K-Nearest Neighbor	36.0	51.3	+15.3
Probabilistic Indexing	64.0	62.7	-1.3
Support Vector Machines	56.7	60.0	+3.3
Maximum Entropy	53.3	60.0	+6.7

Table 7.7: The addition of a first-pass using high-precision/low-recall rules improves the performance of seven of the eight systems tested for the *Politics* image data set.

Table 7.7 shows the results of the eight systems used for this experiment when applied to the *Politics* image data set. Once again, the first column of results shows the overall accuracy of each system on its own, and the second column of results shows the overall accuracy of each system combined with the high-precision/low-recall rules. This time, the addition of the first-pass improves the performance of seven of the eight systems tested, and the performance of one system gets slightly worse. The only system that has overall accuracy decrease (by 1.3%) is the Probabilistic Indexing system, which, on its own, beats all the other systems for this data set by a large margin. The system that improves the most (15.3%) is the K-Nearest Neighbor system, which has the lowest overall accuracy by far, and still has the lowest after the first pass is added, but by less of a margin. The average change in performance over all systems is an improvement in overall accuracy of 6.0%.

As described in Section 3.1.2.5, the test set for the *Politics* image data set contains 150 images. It turns out that the rules corresponding to the words listed in Table 7.4 apply to 59 (39.3%) of these 150 images. In 53 (89.8%) of these 59 cases, the category associated with the word from the list is the correct prediction. Although this sounds quite high, the fact that the Probabilistic Indexing system has its overall accuracy decrease when the first pass is added means that this system does even better for these cases (it can easily be computed that the Probabilistic Indexing system must get 55 of these 59 cases correct). The other seven systems have performance improve, and so these do not do as well as the rules for the 59 documents to which the rules apply.

According to a one-sided χ^2 test, the p-value for the one system that has performance degrade (Rainbow's Probabilistic Indexing system) is 39.9%, which is not statistically significant (i.e. there is about a 40% probability that two equal systems would see a difference this large due to chance for a test set of this size). The p-values for the individual changes in performance for the other systems (all of which have overall accuracy improve) range from 0.01% to 23.2%. Most of these changes are not statistically significant at the 95% confidence level, but all are unlikely to occur by chance. Testing the average improvement for all systems (over all decisions being made) leads to a p-value that is less than 0.01%; in other words, it is extremely safe to say that this is not due to chance. (Again, all of these p-values should probably be computed only based on the examples to which the rules apply, in which case all of the p-values would be lower, and the significance levels would be higher.)

7.5 Research Related to Learning Rules and Combining Systems

The specific criteria used to determine whether or not words are appropriately indicative of categories comes from Riloff (1996; 1999). In her work, she actually uses "augmented relevancy signatures" which are actually phrases, as opposed to individual words, but the two data sets I am using in this chapter are not large enough to support this. Riloff is working with binary text categorization tasks that require a separate YES/NO decision for every document/category pair; her system classifies a document as belonging to a category if and only if one or more of the relevancy signatures associated with the category are present in the document. She is more interested in precision than recall, and her system does not fall back to other techniques when no signature is present.

The idea of automatically creating a list of words that are indicative of particular categories dates back, at least, to the work by Mosteller and Wallace (1963), in which the authors create lists of words indicative of authors for an authorship attribution problem. Mosteller and Wallace only consider high frequency words such as articles and prepositions, because unlike almost all other text categorization tasks, these words make the most difference for authorship attribution. Frequencies of more contextual words depend on content (i.e. the topic, or subject matter, of the document), but frequencies of some very common words such as "an", "of", and "upon" tend to be consistent within the work of a single author. Mosteller and Wallace use a training set to tune parameters for such common words to fit a Poisson or negative binomial distribution, and then they choose words that are more likely to occur with higher frequencies for one author compared to another. This approach is not appropriate for tasks like the one discussed in this chapter, since content words, which are important for most text categorization tasks, do not occur in enough documents with high enough frequencies to allow for a similar analysis.

The idea of examining a list of automatically generated words, classifying a document as belonging to a particular category if one is found, and moving on with the list otherwise, is reminiscent of approaches used by rule learning systems such as Ripper (Cohen, 1995a), which implements a decision list. Similar rules also appear in decision trees (Mitchell, 1997), and one popular system that uses decision trees is C4.5 (Quinlan, 1986; Quinlan, 1993). Rules in these systems can be more complex then the simple "if a particular word w exists in the document, place the document in category c", but you do sometimes see rules of this form. It is well known that when using a decision list to classify a document, the rules tend to get less accurate as you walk along the list. In other words, for cases in which a rule near the top of the list applies, the prediction made using the decision list is very likely to be correct, but for cases in which a rule near the end of the list is used, the prediction is less likely to be correct.

The work I am presenting in this chapter can be viewed as combining a decision list approach with other approaches (although in my case, there is no particular order assigned to the rules in the list). My system starts with the learned simple rules (if a given word is found, categorize the document as belonging to the corresponding category), but instead of walking along a list until reaching a terminal node, it stops at some point and falls back to some other system. This makes intuitive sense, given that the top of the decision list is usually the most accurate part; if a system gets past this point, it is likely that some other system is more accurate for the current document.

In his text categorization survey paper, Sebastiani (2002) refers to the combination of classifiers as a classifier committee. He points out that a committee is characterized by the methods it tries to combine and also by the means of combining them. Strategies that have been used to combine classifiers include majority voting (use the outcome of the majority of categorizers), weighted linear combination (weight each categorizer based on its effectiveness on a tuning set), and dynamic classifier selection (use the result of the categorizer that is most effective on the examples in the tuning set that are most similar to the current document) (Sebastiani, 2002). Research involving the combination of various categorizers include (Larkey and Croft, 1996) and (Li and Jain, 1998). Results have been mixed; Larkey and Croft (1996) show that combining any two of three classifiers (Rocchio, Naive Bayes, and kNN) beats the best individual classifier for their test set, and combining all three does best of all. Li and Jain (1998), however, find that most of their attempts to combine classifiers (Naive Bayes, kNN, decision trees, and their own "subspace method") do not beat the best individual classifiers for a much larger data set.

One other topic related to combining systems is boosting (Schapire, Singer, and Singhal, 1998; Schapire and Singer, 2000). Boosting is an approach that uses a procedure called a *weak learner* to learn many *weak hypotheses*. Each weak hypothesis is a simple and only moderately accurate categorization rule. Boosting algorithms train the weak hypotheses sequentially (as opposed to in parallel) such that each rule is likely to perform well on the examples which are the most difficult to classify by the preceding rules. In this way, the weak hypotheses are combined into a (hopefully) accurate system.

7.6 Concluding Discussion of High-Precision/Low-Recall Rules

The results reported in Section 7.4 surprised me. I was expecting the performance of the NLP Based system to improve, but I was not expecting the performance of the other systems to improve. My thought was that bag of words approaches would already be using the highly indicative words, along with all of the other words. I did not realize that the use of additional words would hurt the bag of words approaches; however, as we saw in the previous chapter, when dealing with these data sets, many of the words in the the first sentences of these captions do not matter, and some are even misleading. The results reported here make it clear that when there is a very indicative word that offers an excellent clue as to the correct category for the image, this word alone performs better than all of the words in the caption including the indicative one. The other words offer more noise than useful information. This approach may therefore generalize and be useful for other text categorization tasks as well. There is nothing about it that is intrinsically designed to work specifically for tasks involving images.

I have not yet attempted to find lists of highly indicative words that would have corresponding high-precision/low-recall rules for either the *Indoor* versus *Outdoor* data set, or for the Events data set. However, I do not believe that this approach would be useful for these categories. The systems I have discussed throughout this thesis already do extremely well for these categories, with accuracy approaching 90% (my BINS system beats 90% for the Events data set when it combines bin-based weights with Naive Bayes weights). Even if we limit ourselves to lists of words that occur in a single category above 90% of the time in the training set, chances are that for the test set, the accuracy of these rules will be something below that, as is the case in this chapter. Meanwhile, the cases for which the rules apply tend to be easier than the average case because of the highly indicative word (e.g. we saw in this chapter that the Probabilistic Indexing system, which achieves a 64.0% overall accuracy for the *Politics* image data set, beats 89.8% for the cases that contain indicative words). So, if we are dealing with categories for which systems already do quite well, it is unlikely that single, highly-indicative words would lead to better performance than all words together.

With both the *Disaster* image data set and the *Politics* image data set, however, the overall accuracy percentage of systems reaches the mid 60's at best, and there is much room for improvement. When dealing with with this kind of performance, a high-precision/low-recall rule based on a single word can achieve better performance for the cases when it applies than using all words. I therefore hypothesize that the technique discussed in this chapter is useful when dealing with difficult categories for which existing systems achieve low overall accuracies. I furthermore have a hunch that this will occur more often when dealing with the categorization of images than when dealing with the categorization of lengthy text documents. The categorization of images is often hard due to less text and categories that may involve determining the focus of an image. Whether or not my hypothesis (or my hunch) is correct will have to be verified by testing the technique with many different data sets.

Other future work involving high-precision/low-recall rules might involve fully automating the process. As things exist today, I have written a script that discovers potential words that are highly indicative of categories. Based on a combination of intuition and statistics, a user can decide which to use and which to prune. Better would be to have this decision made automatically. Different combinations of X and P, as discussed in Section 7.1, have various probabilities of occurring by chance, and perhaps these probabilities, in conjunction with word statistics such as IDF, can be used to keep words that are indicative of categories with very low probability of including words that appear mainly in one category by coincidence.

Finally, the work presented in this chapter fits into the general paradigm of combining systems. In this case, a rule-based approach involving high-precision/low-recall rules is being combined with other systems relying on bag of words approaches. The rules are used whenever they apply, since they are likely to do better in these cases, and the fall back system is used otherwise. This is a bit similar to the dynamic classifier selection strategy for combining system discussed in Section 7.5, since that strategy involves using the classifier that is most likely to do well for the current example; In that case, this is determined by choosing the classifier that performs the best on the examples in the training set that are most similar, whereas in the case of the methodology discussed in this chapter, it is assumed that rules are better when they apply. Perhaps a better idea would be to use a technique such as density estimation (discussed in Chapter 4) to estimate the probability, or confidence level, that the fall back system is correct, and to use the rule only if its expected accuracy is greater than that of the fall back system.

Chapter 8

Low-Level Image Features

Images present us with a type of information that is not available for typical text categorization tasks; namely, low-level image features. In other words, the image itself - or the pixels that form the image when it is stored in a digital format - can be used to predict the appropriate category. This information is not helpful for all categorization tasks. For example, while a human looking at an image may have a very good idea if the picture is related to a *Disaster* or a story on *Politics*, image processing is far from a point where objects can be recognized well enough to make such a determination. However, for certain other tasks, such as the categorization of an image as *Indoor* or *Outdoor*, information such as color and image texture can provide good hints as to the appropriate prediction. This chapter describes my research involving the use of low-level image features to categorize images.[1]

[1]This research has partially been conducted in collaboration with Seungyup Paek and Ana Benitez from the Electrical Engineering Department. Both students have, at times, provided me with information, code, and pointers to relevant material. Earlier efforts combining my text work with Paek's image feature work is discussed in (Paek et al., 1999). Related work of Benitez is discussed in (Benitez and Chang, 2002b) and (Benitez and Chang, 2002a). The specific results I describe in this chapter, however, are all based on my own code, some of which is a reimplementation of code from Paek or Benitez. (I no longer have access to the code from Paek, and the code from Benitez does not run on the specific system I typically use for research.) This chapter is further from my main focus than the others, since I focus on text and not image features; for this reason, the help provided by these students has been particularly valuable.

Of the various data sets I have defined for my own corpus, first described in Section 3.1, I believe that the only one for which low-level image features are useful is the *Indoor* versus *Outdoor* data set. These are abstract categories dealing with the general setting of an image. The other data sets deal with categories involving more specific, semantic meaning, and to use image features for distinguishing between these categories would probably necessitate better object recognition than is currently available. While categories such as *Indoor* and *Outdoor* might, at first, seem somewhat trivial, I have explained in Section 3.1.2.2 some of the reasons that these categories are actually important. Often, they are the first level of an image taxonomy, and (Vailaya et al., 1999b; Vailaya et al., 1999a) present evidence that this matches human intuition. In my own research experience, I have personally noted that the *Indoor* versus *Outdoor* distinction often represents more than just that; for example, in the terrorism domain (in which several researchers in the group were working at the time), *Outdoor* images tended to be at the scene, whereas *Indoor* images tended to be meetings or press conferences. With incentives such as these to motivate research, it is no wonder that I am able to cite several works in Section 8.4 specifically dealing with these categories.

It might be asked why it is important to use image features for categorization tasks involving this particular set of categories when we have already seen throughout this thesis that text can do very well. The accuracy of the text-based systems that have been applied to the *Indoor* versus *Outdoor* data set throughout this thesis has ranged from about 77% to about 87%, and so the best of these systems might serve quite well for applications that make use of these categories. However, it is always nice to do better. I show in this chapter that image features have not performed as well for this data set as text, but that a combination of the two tends to perform better than either alone. Also, categorization based on only image features, while not as accurate as that based on text captions for this particular data set, may be necessary for real world applications when text is not available.

8.1 Color Histograms

Of the several low-level image features that I have come across in my research, the one that I have decided to use for my own implementation of an image feature categorizer is color histograms. There are a few reasons why I have chosen this feature. First, color histograms are relatively simple to understand, and it seems intuitive that they might be indicative of these categories. For example, seeing certain shades of blue at the top of an image might be indicative of sky and, therefore, an *Outdoor* image. Second, I have seen in the literature that color histograms have been successful for *Indoor* versus *Outdoor* categorization (Szummer and Picard, 1998; Paek et al., 1999) and also for similar tasks (Vailaya et al., 1999b; Vailaya et al., 1999a). Finally, it is a relatively easy feature to implement, as is explained in this section.

There are multiple ways to specify color. One is with the RGB (red, green, blue) color space. In other words, each color is specified by its red, green, and blue components. Another possibility is to use the HSV (hue, saturation, value) color space. This space is generally considered more attractive because these components correspond better to human perception of color (Paek et al., 1999; Kerminen and Gabbouj, 1999).

It is possible to convert RGB values to HSV values. Assume that $[r, g, b]$ is defined such that $r, g, b \in [0, 1]$, and you want to obtain the corresponding $[h, s, v]$ values such that $h \in [0, 6]$, $s \in [0, 1]$, and $v \in [0, 1]$. Then the conversion can be performed using the following equations (Kerminen and Gabbouj, 1999; Smith and Chang, 1995; Smith and Chang, 1996b):

$$v = max(r, g, b),\ range = v - min(r, g, b),\ s = range/v \tag{8.1}$$

$$r' = (v - r)/range,\ g' = (v - g)/range,\ b' = (v - b)/range \tag{8.2}$$

$$h = \begin{cases} 5 + b' & \text{if } r = max(r, g, b) \text{ and } g = min(r, g, b) \\ 1 - g' & \text{if } r = max(r, g, b) \text{ and } g \neq min(r, g, b) \\ 1 + r' & \text{if } g = max(r, g, b) \text{ and } b = min(r, g, b) \\ 3 - b' & \text{if } g = max(r, g, b) \text{ and } b \neq min(r, g, b) \\ 3 + b' & \text{if } b = max(r, g, b) \text{ and } r = min(r, g, b) \\ 5 - r' & \text{otherwise} \end{cases} \tag{8.3}$$

Note that the equations break down when r, g, and b all have equal values; in this case, the range of the values is zero, which means that the color is a shade of gray. Once this special case is taken into account, these formulas provide a simple way of converting from the RGB color space to the HSV color space.[2]

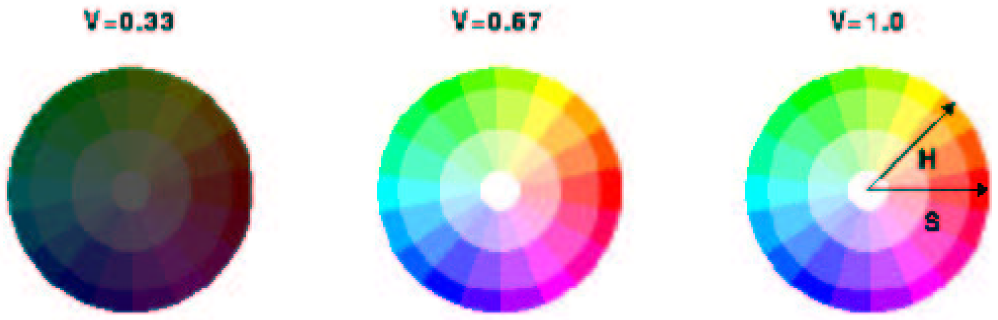


Figure 8.1: The quantized HSV color space, as shown here, allows for 18 hues, three saturations, and 3 values (plus four grays not shown).

Once a color is represented by its hue, saturation, and value, it is, of course, possible to quantize each value. I have followed the model described in several

[2]I have found this to be helpful, since the Perl module Imager, currently available at http://www.eecs.umich.edu/~addi/perl/Imager, provides a method to determine the RGB attributes of a pixel, but does not provide a method to determine the HSV attributes of a pixel.

papers (Smith and Chang, 1995; Kerminen and Gabbouj, 1999; Paek et al., 1999) allowing for 18 possible hues, 3 possible saturations, and 3 possible values, and also allowing for 4 possible grays (based on the shared magnitude of the h, s, and v); the total number of possible quantized HSV representations for a color is this $18 \times 3 \times 3 + 4 = 166$. Figure 8.1, taken from (Smith and Chang, 1996b), demonstrates the 162 possible combinations (not including the four grays). The three circles represent the three quantizations of value; within each circle, the three levels at three separate distances from the center represent the three quantizations of saturation; and within each level of each circle, the 18 components separated at 20 degree increments around the circle represent the 18 quantized hues.

It is now possible to compute a color histogram for an image (or for a region of an image). The pixels of the image are analyzed one at a time, and the HSV attributes of all pixels are determined. All values are quantized, and each pixel is assigned to one of 166 possible slots as explained above. The color histogram of the image is a vector consisting of 166 values, each representing the percentage of pixels that fall into a given slot. Therefore, the sum of the values in the vector for a given image is exactly one. The order of the slots does not matter as long is it is consistent for all images.

8.2 Using Color Histograms to Categorize Images

8.2.1 Input for Machine Learning Techniques

Once color histograms have been computed for images, several of the general, machine learning approaches discussed in Section 2.6 can potentially be applied to train a system using a training set and then predict labels for the images in a test set or for images with unknown labels. Most of the machine learning methods discussed in that section, with the exception of the Rocchio/TF*IDF method, do not need to assume that the features of the input vectors represent textual terms (and even the Rocchio/TF*IDF method can be generalized by using a different weighting scheme appropriate for the new domain). So, for example, a kNN method can be used, in which case the color histogram of a test image is compared with the color histograms of all training images, and the labels of the k nearest training images are used to predict the label of the test image. Or, an SVM method can be used, in which case the color histogram of the test image is mapped to some other feature space, and depending on which side of a hyperplane (determined based on the color histograms of training images) it falls, the test image either is or is not placed in some given category.

While these machine learning methods are general, and do not only apply to text categorization tasks, a specific implementation of any given approach may not be general. For example, my own density estimation system (discussed in Chapter 4), my own BINS system (discussed in Chapter 5), and the systems which comprise the Rainbow package (discussed in Section 2.8) are all text categorization systems that assume textual input. However, the SVM^{light} system, which is also discussed in Section 2.8, is a more general categorization system, which accepts as input any vectors of real values. Therefore, documents that are represented by color histograms can be used as input to this system. In addition, I have implemented a kNN system which also takes input in this general form, and so color histograms can be used for this system as well.

8.2.2 Dividing an Image into Regions

An alternative to computing color histograms based on an entire images is to divide each image into regions and to compute color histograms based on each region of each image (Paek et al., 1999). The final vector for any given image is then the concatenation of the vectors of its individual regions. The reason this might be a good idea is that the presence of some particular color may be more indicative of a category when it appears in a certain region of the image. For example, seeing certain shades of blue at the top of an image may be indicative of sky, and therefore the *Outdoor* category, but the same shades of blue at the bottom of an image may be less indicative of the same category. Vector representations of different images can still fairly be compared to each other as long as each image is broken into regions in the same, consistent manner.

I have performed experiments using my own kNN system and also SVM^{light} under two conditions; Under the first condition, I compute color histograms based on entire images. Under the second condition, I follow the model suggested by Paek (1999) and divide all images into 8 x 8 rectangular regions of equal size. Under this condition, the vector associated with each image therefore has $8 \times 8 \times 166 = 10,624$ values. This, of course, slows down the experiments dramatically. To perform a complete run including training and testing on one of the department's faster machines, my kNN system takes minutes when histograms are computed based on entire images, but hours when images are divided into regions. SVM^{light} takes seconds when histograms are computed based on entire regions, and minutes when images are divided into regions. Also, the division of images into regions itself has been pre-computed for all images, and this process takes several hours.

Figure 8.2 shows a sample image broken down into 8 x 8 regions of equal size. This is not an image from the *Indoor* versus *Outdoor* data set; rather, it is a photograph of me and three colleagues at a nearby McDonald's taken a few years ago. I am the third person from the left. Imagine that a categorization system that uses low-level image features has to predict whether this is an *Indoor* or an *Outdoor* image. Note the shade of blue of my shirt; if this is seen at the top of

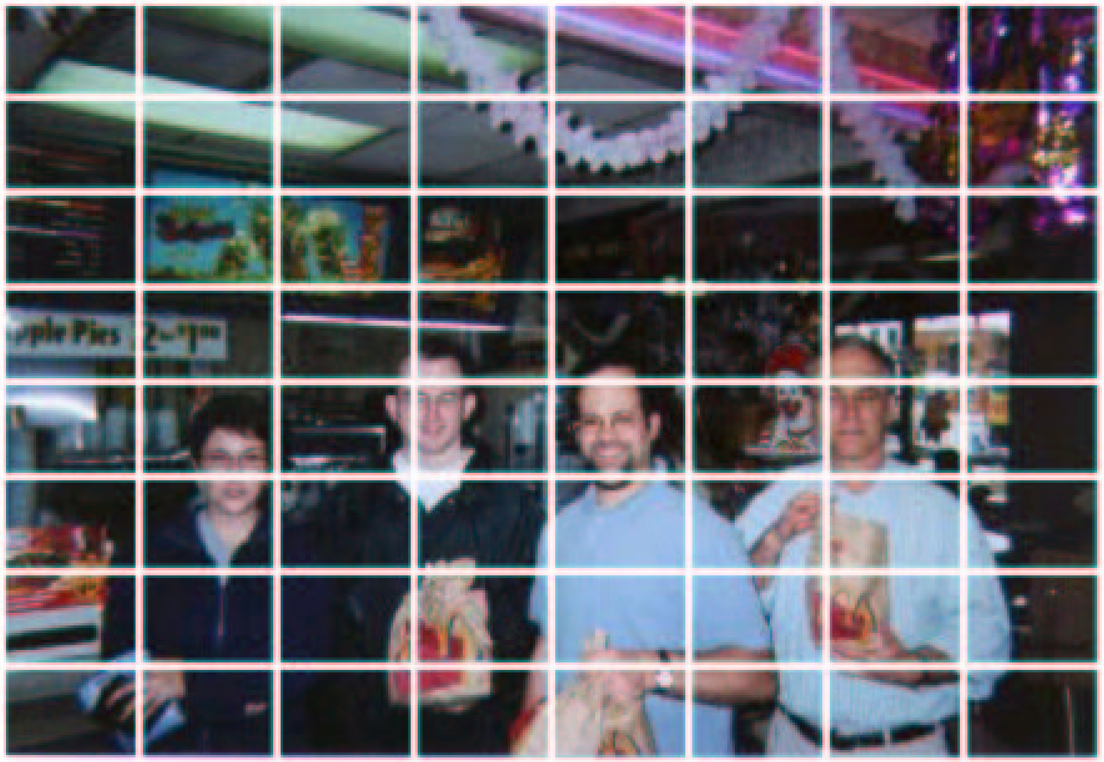


Figure 8.2: This is an example of an image being divided into 8 x 8 rectangular regions of equal size. It is possible that for specific categorization tasks, certain colors matter more in some regions than in others.

an image, it might be indicative of sky, but at the bottom of the image, it is not; therefore, it might be helpful to divide this image into regions. (It turns out that the SVM^{light} system, which is the system that performs better for these categories and is used for the experiments combining image features with text as discussed in Section 8.3, gets this image wrong anyway. However, it is closer to making the correct decision when the image is broken down into regions; in other words, the confidence of the system's incorrect *Outdoor* prediction is lower when the image's color histogram is based on regions than when the image's color histogram is based on the entire image.)

8.2.3 Results for the Indoor versus Outdoor Data Set

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
Image Features			
K-Nearest Neighbor (whole images)	74.8	30.9	84.6
K-Nearest Neighbor (image regions)	71.7	10.0	83.2
SVM^{light} (whole images)	74.8	54.8	82.5
SVM^{light} (image regions)	78.0	58.1	85.1
My text systems			
BINS (COMBO #2)	87.2	78.0	91.0
BINS (COMBO #1)	86.1	76.2	90.2
BINS (always use bin)	85.8	75.1	90.1
Naive Bayes	84.5	70.9	89.4
Rocchio/TF*IDF	80.7	69.9	85.7
Density Estimation	86.1	73.7	90.5
K-Nearest Neighbor	82.7	65.8	88.4
Rainbow systems			
Naive Bayes	85.4	73.5	89.9
Rocchio/TF*IDF	84.5	73.2	89.1
K-Nearest Neighbor	77.8	65.3	83.6
Probabilistic Indexing	86.3	78.1	90.0
Support Vector Machines	82.0	66.9	87.7
Maximum Entropy	84.5	70.9	89.4

Table 8.1: SVM^{light} with image regions does the best of the systems using color histograms, although still not as good as most of the text based systems.

Table 8.1 shows the results when systems use image features (i.e. color histograms) to predict categories. With both SVM^{light} and my own kNN system, I have repeated experiments using color histograms based on whole images, and also using color histograms based on image regions. (When testing the kNN system, I have also tried out multiple values of k and I am reporting only the best results.) For comparison, I have also presented the results of the various text based systems discussed throughout the thesis. The overall accuracy of text based systems ranges from 77.8% (for Rainbow's kNN system) to 87.2% (for one variation of my BINS system). For further comparison, a baseline system which chooses the larger category (*Outdoor*) every time achieves a 71.2% overall accuracy while a system that guesses randomly would achieve approximately a 50% overall accuracy.

When color histograms are based on whole images, both my kNN system and SVM^{light} achieve an overall accuracy of 74.8%. When color histograms are based on image regions, the overall accuracy of the kNN system falls to 71.7%, and according to a one-sided χ^2 test, the p-value for this change is 7.9%; the overall accuracy of SVM^{light}, on the other hand, rises to 78.0%, and according to a one-sided χ^2 test, the p-value for this change is 6.0%. So neither of these changes are significant at the 95% confidence level, but both are significant at the 90% confidence level. Comparing the two systems directly when color histograms are based on image regions (i.e. comparing SVM's 78.0% overall accuracy to kNN's 71.7% overall accuracy), a one-sided χ^2 test indicates a p-value of 0.2%, which is very statistically significant.

Note that the kNN system does not perform as well as any of the text based systems and barely beats the baseline, especially when images are divided into regions. This is likely due to a drawback of kNN systems, which is that they weight all features equally in their standard form. For text categorization purposes, this is often avoided by appropriately weighting terms. For example, if TF*IDF weights are used, as explained in Section 2.4.2, words that are unimportant (at least, for most categorization tasks) have very low weights due to the IDF component of the weighting scheme, and whether such a word occurs no times, one time, or many times will not make much of a difference when computing the distance between two vectors. However, when using color histograms to represent documents, each weight represents the percentage of time that a quantized color shows up in an image (or in a region of an image); all values are weighted on an equal scale, even though many do not matter. Thus, there is a lot of noise, and this throws off results. The effect for the particular task being considered in this chapter is that most of the nearest neighbors of almost all test images are *Outdoor* images, simply because this is the larger category. Very few images are thus predicted by the system to be *Indoor* images, and while the precision for the *Indoor* category is quite high, the recall for the *Indoor* category is very low, thus explaining the low F_1 measures achieved by the kNN system for this category. (These evaluation metrics are described in Section 2.7.)

An SVM system does not suffer from this drawback. In other words, an SVM system can distinguish important features (i.e. colors) from non-important features. This helps explain why the SVM system performs better than the kNN system for the task being considered, especially when images are divided into regions. The regions greatly increase the number of features per document (there are 10,624 instead of 166, as explained in Section 8.2.2); for a kNN system, this means more noise, but for an SVM system, it means more good and specific information to use. Providing color histograms to SVM^{light} after dividing images into regions leads to a performance of 78.0%, which is significantly better than the baseline, and even beats one (but still not most) of the text based systems. Depending on the application and the needs of users, this might be good enough to use for the *Indoor* versus *Outdoor* task when text is not available.

Of course, it might be possible to achieve better results by using additional low-level image features, such as edge direction histograms and Daubechies' wavelets, in addition to color histograms. Research involving these features is discussed is Section 8.4. However, this is not the focus of my own research. I am primarily interested in text, and the purpose of my work with image features has been to see if I could improve text based systems by combining results from an image feature system. For this purpose, it turns out that the results using SVM^{light}

with color histograms based on image regions is adequate, as is explained in the next section.

8.3 Combining Text and Color Histograms to Categorize Images

The overall accuracy of text based systems that I have tested for categorization involving the *Indoor* versus *Outdoor* data set ranges from 77.8% to 87.2%. In the previous section, I have shown that using low-level image features, and in particular color histograms, as input to a publicly available SVM system achieves an overall accuracy of 78.0%. Clearly, this is enough to improve the performance of the lowest text-based system (since replacing all predictions with the new predictions would lead to a small improvement), but what is not clear is whether we can obtain an improvement for most or all text based systems. Can the predictions of an image based system that obtains an overall accuracy of 78.0% improve the accuracy of a text based system that, on its own, does almost 10.0% better?

In Section 8.4, I discuss efforts of other researchers in the field to combine results of multiple systems with the goal of improving accuracy. In the literature, results of such endeavors have been mixed. However, almost all previous efforts have involved combining results of multiple text based systems. Even though the various systems are using different approaches to make their decisions, they are all basing decisions on the same information. Therefore, there is reason to believe that the various systems should find the same documents easy or hard to predict. On the other hand, the research discussed in this section involves the combination of a system that bases its predictions on image features (specifically color histograms) with systems that base their predictions on text. There is no reason to believe that an image which has clearly indicative text should also have clearly indicative colors, and vice versa. So, even though the system using image features does not do as well as most of the systems using text, it is quite possible that there are certain images for which the confidence of the image feature system is high enough such that its prediction is more likely correct than that of any text based system. When this occurs, it might be better to use the prediction of the image feature system, and to fall back to a text based system otherwise.

8.3.1 Evaluating Various Levels of Confidence

As explained in Section 2.6.2.3, when an SVM system is given a vector representing a document, that vector is mapped to some feature space, and in that feature space the system checks on which side of a hyperplane the mapped vector falls. For this particular task, if the vector falls on one side of the hyperplane, the system predicts that the image is an *Indoor* image, and if the vector falls on the other side of the hyperplane, the system predicts that the image is an *Outdoor* image. In order to combine these predictions with that of other systems, it is helpful to have some sort of confidence measure of each prediction. For this purpose, the distance between each mapped vector and the deciding hyperplane can be used as the level of confidence. This only works well if it is true that mapped vectors which are further from the hyperplane actually do have a greater chance of leading to a correct prediction.

Table 8.2 shows the performance for various levels of confidence of SVM^{light} using color histograms based on image regions. The first column represents the confidence cutoff being considered in steps of 0.5 (i.e. only images whose mapped vectors fall at least this distance from the hyperplane are considered). The final column shows the number of images that qualify. It turns out that no test images in the *Indoor* versus *Outdoor* data set are predicted with a confidence greater than or equal to 3.5. If we lower the cutoff to 3.0, two images qualify; both are *Outdoor*

Distance Cutoff	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %	% of Images Above Cutoff
3.5	—	—	—	0.0
3.0	100.0	—	100.0	0.4
2.5	87.5	0	93.3	1.8
2.0	92.3	50.0	95.8	5.8
1.5	94.4	77.8	96.8	16.0
1.0	91.0	69.6	94.7	34.1
0.5	84.6	60.7	90.4	70.1
0.0	78.0	58.1	85.1	100.0

Table 8.2: The further that an image's mapped vector is from a hyperplane, the greater the chance that the prediction of the system is correct.

images, and both are predicted correctly. As we continue to lower the cutoff, more and more images qualify. Obviously, if we lower the cutoff all the way to zero, all 445 images in the test set qualify, and the overall accuracy for all these images is 78.0% as has already been reported in Section 8.2.3.

Typically, we would expect that the higher the confidence, the greater the probability that the system's prediction is correct, and therefore we expect to see the overall accuracy of the system go down as the cutoff for the confidence is decreased. In general, this holds true here. There are two exceptions at the top of the table; the overall accuracy increases as the cutoff is lowered from 2.5 to 2.0 or from 2.0 to 1.5. However, the number of images with confidences above the 2.0 cutoff is still small, and so these overall accuracies are probably not accurate.

One very promising result is that for certain cutoffs (all those 1.0 or greater), the accuracy for the image based system is greater than the overall accuracy of any text based system (when applied to the entire test set). While it is possible that one or more of the text based systems may beat the image based system if only those images that beat the cutoff are considered, this does not seem likely, since there is no intuitive reason to believe that images with very indicative colors should also have associated text that is more indicative than average text. This observation suggests as an approach similar to the one I describe in Chapter 7 to combine high-precision/low-recall rules with other text based systems. The idea is to use the prediction of the image based system only for those cases in which it is likely to do better than a text based system (i.e. for images with appropriately high confidence), and to fall back to one of the text based systems for all other cases. In other words, a prediction is made using SVM^{light} and color histograms based on image regions for all test images. For those images for which the prediction is made with a confidence greater than a particular cutoff, the prediction stands; for the rest of the images, a text based system is used instead.

To me, the most promising lines of the table are the lines representing the cutoffs of 1.5 and 1.0. At these levels, the overall accuracy of the system is still quite impressive (above 90.0%, and significantly better than any of the text based systems achieve on the entire test set), while applying to enough images such that we might expect to see a nice improvement to the text based system with which the image based system is being combined. It might be pointed out that since the overall accuracy decreases as the cutoff is dropped from 1.5 to 1.0, the higher cutoff might be better, since it might be the case that the accuracy for the specific images added (i.e. the ones with confidence between 1.0 and 1.5) might be below the accuracy of text based systems. However, it can easily be calculated that if we only consider the images with cutoffs in between 1.0 and 1.5, there are 84 such images, and 88.1% of them are categorized correctly. This is still a better accuracy than any text based system achieves for the entire test set, and so we are likely to see additional improvement as we include these new images. However, several of the text based systems achieve overall accuracies reasonably close to 88.1%, so there is a reasonable chance that they may do this well for these specific images, and the decision between these two cutoffs is not entirely clear.

Interestingly, if we move one step further and look only at the 157 images with confidence between 0.5 and 1.0, only 78.3% are categorized correctly. This is below the overall accuracy of most of the text based systems. Finally, if we only consider the 133 images for which the confidence is less than 0.5, the accuracy for these images is only 62.4%. This falls way below the overall accuracy of any text based system and even way below the baseline based on choosing the larger category every time (although it is still better than random guessing).

8.3.2 New Results for the Indoor versus Outdoor Data Set

System	Overall Accuracy % with Specified Cutoff							
	3.5	3.0	2.5	2.0	1.5	1.0	0.5	0.0
My systems								
BINS	87.2	87.2	87.2	87.4	87.9	87.2	84.3	78.0
Rainbow systems								
Naive Bayes	85.4	85.4	85.4	85.6	85.8	85.6	83.8	78.0
Rocchio/TF*IDF	84.5	84.5	84.5	84.5	84.7	84.5	83.6	78.0
K-Nearest Neighbor	77.8	78.0	78.0	78.9	80.9	82.3	83.2	78.0
Probabilistic Indexing	86.3	86.3	86.3	86.1	87.0	86.7	86.7	78.0
Support Vector Machines	82.0	82.0	82.0	82.9	82.9	82.9	82.0	78.0
Maximum Entropy	84.5	84.5	84.7	85.2	85.2	84.7	83.2	78.0

Table 8.3: With an appropriate confidence cutoff for using the predictions based on image features, gains are likely for all text based systems.

Table 8.3 shows the results combining image features with text for the *Indoor* versus *Outdoor* data set. As described, the results of SVM^{light} using color histograms based on image features are used if the confidence of the system's prediction is greater than the specified cutoff; otherwise, we fall back to a text based system. As you can see, I have tried using confidence cutoffs from 0 to 3.5 with steps of 0.5, and for the fall back systems, I have tried using BINS (with its best settings) and the Rainbow systems. As explained in Section 8.3.1, the confidence is never greater than 3.5 for this data set, and so the first column of results represents the results of the text based systems. The confidence is always non-negative (since it is really just the distance of a mapped vector from a hyperplane), and so the final column of results, using 0.0 as the cutoff, represents the results of using only image features, and is not affected by the particular text based, fall back system used.

The first thing to note is that with a cutoff for confidence of 1.0 or better, the combined result of text and image features is almost always at least as good as the better of the two individual systems. (All text systems except Rainbow's K-Nearest Neighbor system beat the image feature system, and the kNN system only loses by a very small margin.) The only time performance goes down with a confidence of 1.0 or greater is when the fall back system is Rainbow's Probabilistic Indexing system and a cutoff of 2.0 is used; in this case, the combined performance is marginally lower than the text based system's performance.

Remember from Section 8.3.1 that the two most promising cutoffs are 1.5 and 1.0. Looking at the combined results using these two cutoffs, we see that the combined result is always at least as good as the better of the two individual systems. With a cutoff of 1.5, the combined result is always better than the two individual systems, and with a cutoff of 1.0, the combined result is tied with the text result for two cases (Rainbow's Rocchio/TF*IDF system and my BINS system) and better than the two individual systems for the other five cases. The average performance of the seven individual text based systems (when used alone) is 84.0%. The average performance when combined with image features (using a cutoff of either 1.5 or 1.0) is 84.9%. According to a one-sided χ^2 test, this change in the average performance of all systems (over all decisions being made) leads to a p-value of 8.6%. In other words, we can not say the change is statistically significant at the 95% confidence level, but it is significant at the 90% confidence level. (If only documents for which predictions are based on image features were used to determine significance, the p-values would be lower and the significance higher, since the same number of examples change from wrong to right based on a smaller set.)

For most text systems, the improvement gained by combining image features is not large. Generally, the best improvement seen for an individual system is slightly under 1%, and for Rainbow's Rocchio/TF*IDF system, the best improvement seen is only 0.2%. According to a one-sided χ^2 test, these changes are not statistically significant at the 95% confidence level. The best improvement seen is for Rainbow's K-Nearest Neighbor system, in which case the best improvement seen occurs with a cutoff of 0.5, and the improvement is over 5%. The improvement for this system is still 4.5% when a cutoff of 1.0 is used. Both of these improvements are statistically significant (the p-values are 0.4% and 1.4%, respectively). The kNN system has the lowest performance of all text based systems, and approximately the same performance as using image features alone.

8.3.3 The Potential for Further Improvement

These results are promising. It seems that even when image features alone perform well under text alone, a small improvement can still be expected when the appropriate confidence cutoff is used. When image features alone perform about as well as text alone, on the other hand, a large improvement might be expected, as is the case for Rainbow's kNN system. As discussed at the end of Section 8.2, using image features to categorize images has not been the focus of my research; I have been primarily concerned with text. I have implemented code to divide images into regions and to compute color histograms, and I took advantage of a general, public categorization system (SVM^{light}) to get results using this feature. However, my main purpose for doing this was to combine these results with text based results. It is quite possible that using additional image features such as edge direction histograms and Daubechies' wavelets would lead to better results based on image features alone. (I discuss some research using features such as these in Section 8.4.) If these results approach the results of other text based systems, it is quite possible that combining results of image and text would achieve significant gains, as is already the case for the kNN system.

In addition, there are probably better ways to combine predictions of image based systems and text based systems. The method I have described in this section only uses the confidence of the image based system to decide which prediction to use. Ideally, the confidence of both systems should be used. In other words, given the predictions of an image based system and a text based system, whichever prediction has the higher confidence should be used. It is quite possible that for certain images for which the image based prediction is currently being used due to a high confidence, the text based confidence may be even higher; similarly, it is possible that for certain images for which the image based confidence is low, the prediction should still be used if the text based confidence is even lower. The reason I have not already tested this method of combining systems is that different systems do not compute confidence in the same way (if at all), and so confidence measures are not directly comparable. However, density estimation, described in Chapter 4, provides a way to compute confidence measures for all of these systems. By applying density estimation to the output of all of these systems, results can be converted to probabilistic confidence measures that are directly comparable. If these confidence measures are accurate, this may lead to a method of combining results that works better than the one described in this section, and if so, further gains might be achieved. (Appendix H provides evidence that the confidence measures are accurate for the experiments applying my density estimation to the *Indoor* versus *Outdoor* data set and the Events data set.) I leave this for future work.

8.4 Research Related to Image Features and Combining Systems

There is actually a history of image processing research dealing specifically with *Indoor* versus *Outdoor* distinction. The work of Szummer and Picard (1998) represents an attempt to categorize consumer photographs from vacations as *Indoor* or *Outdoor* using low-level features such as color histograms. The research discussed in (Paek et al., 1999) concerns a colleague's efforts involving the use of low-level image features including color histograms and edge direction histograms for categorization involving the same categories, in addition to my own work using text for this task and also an early attempt to integrate the two two approaches together.

The work discussed in (Vailaya et al., 1999b) and (Vailaya et al., 1999a) involves the use of similar low-level image features to divide *Outdoor* vacation images into *City* images and *Landscape* images, and they further consider dividing the *Landscape* images into the categories *Sunset*, *Forest*, or *Mountain*. At all levels (*Indoor* versus *Outdoor*, *City* versus *Landscape*, and *Sunset* versus *Forest* versus *Mountain*), the authors recognize the existence of an *Other* category, which has also occurred throughout this thesis (although I have not considered the same sets of categories as they do). In their work, all sets of categories were motivated by experiments with human subjects who evaluated 171 vacation images and were asked to group the images into meaningful categories. This helps provide support that the *Indoor* versus *Outdoor* distinction is a pragmatic one.

The research described in (Wang et al., 1997) involves the use of low-level image features such as Daubechies' wavelets and color histograms to classify images as *objectionable* or *benign*, where objectionable images are those that are pornographic. The authors discuss the potential of using text in addition to image features, but I have not found later work by them that actually attempts to do this. Earlier work described in (Fleck, Forsyth, and Bregler, 1996) involves the use of low-level image features for the related task of detecting naked people in images. These research endeavors have the obvious application of filtering objectionable images from children in mind.

Many systems have been developed for image retrieval (as opposed to categorization) based on image features. Some of these that I have come across in my research include QBIC (Niblack et al., 1993), Photobook (Pentland, Picard, and Sclaroff, 1994), Foureyes (Picard and Minka, 1995), VisualSeek (Smith and Chang, 1996c), WebSEEk (Smith and Chang, 1997), and MARS (Hehrotra et al., 1997). Of these, only WebSEEk also uses text. Benitez and Chang (2002b; 2002a) discuss the use of both text and low-level image features to cluster images. The potential for using these clusters to aid in categorization is discussed but not implemented.

Although it is not common in the literature, there have been some prior efforts in the computer vision literature that involving matching pieces of text associated with an image to regions of the image. Srihari (1995) has developed a system called Piction, which matches names in image captions to faces in the image for the purpose of retrieval. The work discussed in (Duygulu et al., 2002) involves the use of the expectation maximization (EM) algorithm and a process analogous to learning a lexicon from aligned bilingual text in order to automatically annotate regions of images with provided keywords.

I have already discussed research related to the combination of systems in Section 7.5, since the work in that chapter can be viewed as combining decision rules with other systems. I reiterate that information here, since this work also relates to the research discussed in this chapter. In his text categorization survey paper, Sebastiani (2002) refers to the combination of classifiers as a classifier committee. He points out that a committee is characterized by the methods it tries to combine and also by the means of combining them. Strategies that have been used to combine classifiers include majority voting (use the outcome of the majority of categorizers), weighted linear combination (weight each categorizer based on its effectiveness on a tuning set), and dynamic classifier selection (use the result of the categorizer that is most effective on the examples in the tuning set that are most similar to the current document) (Sebastiani, 2002). Research involving the combination of various categorizers include (Larkey and Croft, 1996) and (Li and Jain, 1998). Results have been mixed; Larkey and Croft (1996) show that combining any two of three classifiers (Rocchio, Naive Bayes, and kNN) beats the best individual classifier for their test set, and combining all three does best of all. Li and Jain (1998), however, find that most of their attempts to combine classifiers (Naive Bayes, kNN, decision trees, and their own "subspace method") do not beat the best individual classifiers for a much larger data set. Of course, the work described in this chapter is different, in that I am combining the results of text based systems with the results of a system that is basing its predictions on low-level image features. The only previously published work of which I am aware discussing this approach is (Paek et al., 1999), which I have already mentioned earlier in this section.

8.5 Concluding Discussion of Image Features

The content of this chapter is uniquely applicable to images. The rest of the thesis so far has been more general. The majority of this thesis has discussed the categorization of images using associated text (primarily captions). At times, standard text categorization techniques can be applied directly with good results. For certain categories (e.g. those that deal with focus, perspective, or point of view), more advanced NLP techniques are necessary to achieve optimal performance. I have shown an example of this in Chapter 6, and I also explain in that chapter why I believe that this is more likely to occur when categorizing images as opposed to full-length text documents; however, it is possible that similar categories involving text-only documents might also require deeper linguistic processing. In this chapter, on the contrary, I am specifically dealing with low-level image features that are simply not available for most text categorization tasks. It turns out that they are helpful for the categorization of images, at least when dealing with the categories *Indoor* and *Outdoor*.

The first part of the chapter discusses a specific low-level image feature, namely color. I have described how images can be represented by color histograms, either based on an entire image at once, or based on the regions that comprise an image. It turns out that computing color histograms after dividing images into 8 x 8 rectangular regions of equal size, and providing this input to an SVM system, leads to reasonable performance for *Indoor* versus *Outdoor* categorization. Although the performance is not as good as most text systems, it is still much better than baseline performance, and thus this might be useful when text is not available. Furthermore, if only images for which the confidence is above some appropriate cutoff are considered, the accuracy for these images is above that of all text based systems I have tested, and this has led to experiments that combine the results of text and image based systems.

The *Indoor* versus *Outdoor* data set is interesting, because reasonable performance can be achieved for this data set using either text or image features. This has allowed me to experiment with the combination of results based on entirely separate features. In the text categorization literature, research involving the combination of systems has had mixed results, at best, but that is probably because researchers have been combining systems that have based their decisions on the same features. Even though different approaches are used by different systems, if they are basing their decisions on the same features, it is likely that they do well on approximately the same documents. On the other hand, there is no reason to believe that images for which text is particularly indicative of the correct category should also have colors (or other low-level image features) that are particularly indicative, and vice versa. In other words, it is quite possible that the images for which an image based system is most confident (and therefore most likely to predict accurately) are not the same images for which text based systems are the most confident. This shows promise for combining the results of such systems.

I have discussed one method of combining results of an image based system with results of text based systems. This method usually improves results (i.e. the results of the combination are generally better than the results of either text or image features alone), and rarely does it make results worse. For the one case in which the image based system performs nearly as well as the text based system (Rainbow's K-Nearest Neighbor system), the improvement is very significant. I believe that by incorporating additional image features (e.g. edge direction histograms), the results using image features would likely improve, and additional gains would be achieved. Also, a better method of combining results that compares the confidence of the image based system with the confidence of the text based system (perhaps using density estimation) should lead to a better choice of which prediction to use, and therefore improve performance further.

Chapter 9

Newsblaster

The World Wide Web has inarguably changed the way that many people live day to day. One of the many benefits of the web is that it provides many on-line news sources that are updated whenever a new story is available. Such sites include CNN, Reuters, Fox News, and many others. For some people, these sites have replaced printed newspapers and television as their main source of news, and for others, they provide an additional source of news. There are so many such news sites and so many articles posted every day that it is impossible to read them all.

Newsblaster (McKeown et al., 2002) is a publicly accessible system that has been developed at Columbia University to help users find and browse the news that is of the most interest to them.[1] Newsblaster also showcases the research of many people in the Natural Language Processing (NLP) research group at Columbia University. The system automatically collects, clusters, categorizes, and summarizes news from several sites on the web, and it provides users with a user-friendly interface to browse the results. Newsblaster utilizes my own research in two ways; news stories are automatically categorized into sections similar to those in newspapers and other news websites, and images are categorized

[1]For legal reasons, the name of Newsblaster will soon have to change; as of the time I am writing this thesis, the new name has not yet been decided.

with labels that help to provide users efficient browsing. This chapter describes Newsblaster with a focus on my own contributions to the system. Newsblaster can currently be accessed on the web at http://newsblaster.cs.columbia.edu or http://www.cs.columbia.edu/nlp/newsblaster.

There are several pragmatic benefits of Newsblaster. Rather than traverse many different news sites to find news of interest, a user can turn to Newsblaster, which has already explored the various sites. Automatically generated summaries can then help users determine which stories are important to them. If they want to learn more, Newsblaster provides links to the original articles, so a user can read all of the articles pertaining to a given story. For reasons such as these, the Newsblaster system has already caught the attention of the press and public. A recent analysis indicates that Newsblaster receives tens of thousands of hits every day. News agencies that have written articles about Newsblaster include The New York Times, USA Today, and Slashdot. A list of press articles about Newsblaster can be found at http://www.cs.columbia.edu/nlp/newsblaster/press.html.

9.1 Description of the Newsblaster System

9.1.1 Crawling the Web and Gathering News

Newsblaster is a fully automatic system that currently updates itself once a day (plans are currently underway to have Newsblaster increment itself incrementally throughout the day whenever it encounters new news). The first phase of Newsblaster consists of crawling the web, starting at specific, popular news sites such as CNN, Reuters, and Fox News. At the time I am writing this, there are 18 sites that are explored, and the list of sites is displayed at the top of the main Newsblaster page. As of now, all of the sites are in English, but there is work underway to also explore sites in other languages. Links are followed up to a given depth (currently four), and a set of manually created rules is used to determine what, if anything, on each page appears to be a news article.[2] When a news article is found, additional manually created rules are used to find images that relate to the article (as opposed to other images such as advertisements), and if such images are found, more rules are used to find captions for these images, if available. The precision and recall for all these sets of rules appear to be very high.

9.1.2 Detecting Groups of Articles on the Same Story

Once crawling of the websites is complete, articles are clustered into groups of related articles. As mentioned in Section 2.3, clustering, unlike categorization, does not have predetermined categories; rather, documents that are similar to each other are grouped together. The clustering algorithm used is that described in (Hatzivassiloglou, Gravano, and Maganti, 2000), and parameters are set such that each cluster is likely to contain a group of articles describing the same news story. Only clusters that contain at least some specified minimum number of articles (currently four) are presented in the main Newsblaster interface. When the first stage of clustering is complete, a second stage of clustering with different parameter settings cluster the original clusters into "superclusters", such that each supercluster consists of a group of clusters on stories that are related to each other. Typically, thousands of articles are discovered, but of these only hundreds make it into clusters. Many of these clusters contain close to the minimum allowed number of articles, but often there are some that contain as many as a few dozen. Most superclusters contain exactly one cluster, but usually there are a few that contain more than one.

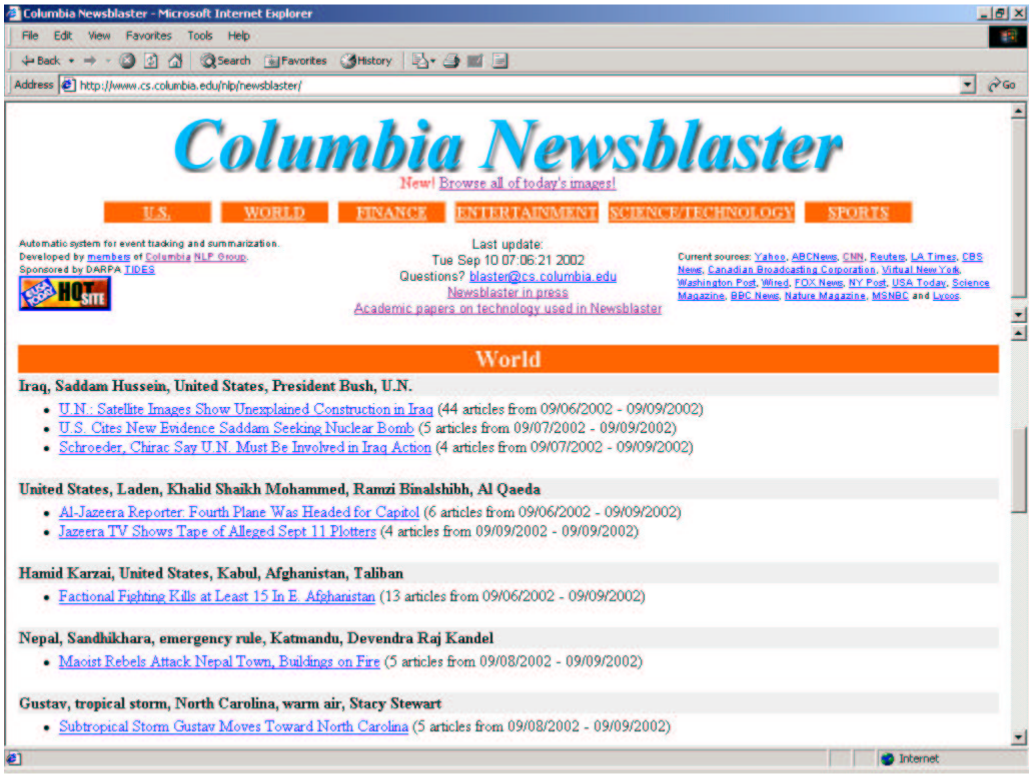
[2]These rules seem to work very well for the sites we are currently exploring, but I am aware that work is currently underway to learn rules for each site so as to do better with multilingual news sources.

9.1.3 Categorizing News Stories into Useful Sections

Before superclusters are displayed to the user, they are categorized into sections that are typical of newspapers and on-line news sites. These sections are currently *U.S. News*, *World News*, *Finance*, *Entertainment*, *Science/Technology*, and *Sports*. The training set consists of 2,869 articles from older Newsblaster runs. I personally labeled the high majority of these articles, although some have come as suggestions from other people as is described in Section 9.2. Appendix P discusses an attempt that had been made to generate a training set for these categories automatically, but the analysis in this appendix shows that the effort did not achieve results as good as a manually created training set. The system that Newsblaster uses to categorize superclusters is my BINS system (discussed in Chapter 5). Categorization is first applied to individual articles, and each supercluster is then assigned to whichever section contains the most articles in the supercluster (ties are broken by choosing the category with the highest sum of scores over all articles in the supercluster).

Figure 9.1 shows a section of the main Newsblaster interface from September 10, 2002. The frame at the top of the page contains general information about Newsblaster, links to pages including papers and articles written about Newsblaster, and links to each of the categories for superclusters. There is also now a link to an image browsing interface (discussed more later in this chapter). The bottom frame of the page as shown has been scrolled down to the top of the *World News* section. The top supercluster in this section on this day consists of three clusters, the next supercluster consists of two clusters, and the rest each contain one cluster. As you can see, a title is displayed for each cluster, as well as the number of articles in the cluster and the range of dates of those articles.[3] In addition, representative

[3]Titles of individual articles are found using regular expressions, and when that fails, the first sentence of the article is used instead; Currently, the longest title of the articles in a cluster, not including those that use first sentences, is used as the title of the cluster, but more advanced methods are being considered.


Columbia Newsblaster - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://www.cs.columbia.edu/nlp/newsblaster/
Go
Internet

Columbia Newsblaster

New! Browse all of today's images!

U.S. WORLD FINANCE ENTERTAINMENT SCIENCE/TECHNOLOGY SPORTS

Automatic system for event tracking and summarization.
Developed by members of Columbia NLP Group.
Sponsored by DARPA TIDES

Last update:
Tue Sep 10 07:06:21 2002
Questions? blaster@cs.columbia.edu
Newsblaster in press
Academic papers on technology used in Newsblaster

Current sources: Yahoo, ABCNews, CNN, Reuters, LA Times, CBS News, Canadian Broadcasting Corporation, Virtual New York, Washington Post, Wired, FOX News, NY Post, USA Today, Science Magazine, BBC News, Nature Magazine, MSNBC and Lycos.

World

Iraq, Saddam Hussein, United States, President Bush, U.N.
- U.N.: Satellite Images Show Unexplained Construction in Iraq (44 articles from 09/06/2002 - 09/09/2002)
- U.S. Cites New Evidence Saddam Seeking Nuclear Bomb (5 articles from 09/07/2002 - 09/09/2002)
- Schroeder, Chirac Say U.N. Must Be Involved in Iraq Action (4 articles from 09/07/2002 - 09/09/2002)

United States, Laden, Khalid Shaikh Mohammed, Ramzi Binalshibh, Al Qaeda
- Al-Jazeera Reporter: Fourth Plane Was Headed for Capitol (6 articles from 09/06/2002 - 09/09/2002)
- Jazeera TV Shows Tape of Alleged Sept 11 Plotters (4 articles from 09/09/2002 - 09/09/2002)

Hamid Karzai, United States, Kabul, Afghanistan, Taliban
- Factional Fighting Kills at Least 15 In E. Afghanistan (13 articles from 09/06/2002 - 09/09/2002)

Nepal, Sandhikhara, emergency rule, Katmandu, Devendra Raj Kandel
- Maoist Rebels Attack Nepal Town, Buildings on Fire (5 articles from 09/08/2002 - 09/09/2002)

Gustav, tropical storm, North Carolina, warm air, Stacy Stewart
- Subtropical Storm Gustav Moves Toward North Carolina (5 articles from 09/08/2002 - 09/09/2002)

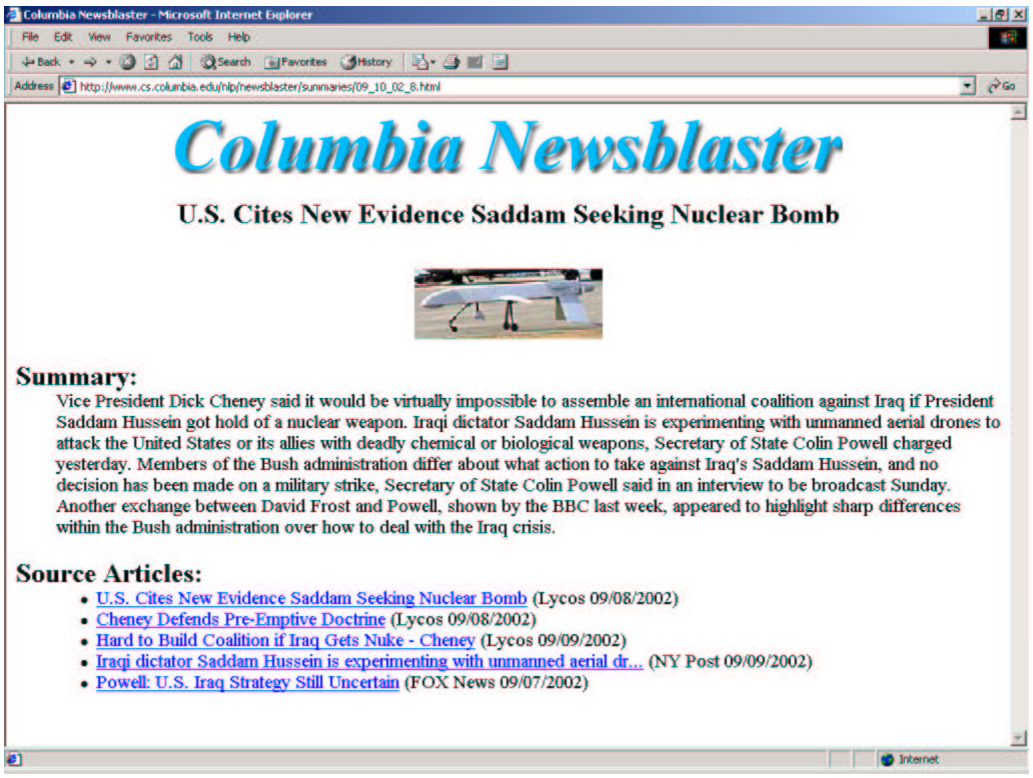
Figure 9.1: The main Newsblaster interface currently displays superclusters in six different categories, including keywords for each supercluster and titles for each cluster.

keywords are displayed for each supercluster, currently selected by a system called Nominator (Wacholder, Ravin, and Choi, 1997).

9.1.4 Presenting Summaries and Appropriate Images

When a user clicks on the title of a cluster, an automatically generated summary of the articles contributing to the cluster is displayed along with links to the original articles and any appropriate images that have been found. Newsblaster uses two separate summarizers for different clusters depending on the type of documents in the cluster as determined by a Router (McKeown et al., 2001). One of the systems, MultiGen (Barzilay, Elhadad, and McKeown, 2002; Hatzivassiloglou et al., 2001;

Barzilay, McKeown, and Elhadad, 1999; McKeown et al., 1999), is used when there is a high enough degree of similarity between the articles in a cluster. The other system, DEMS (Schiffman, Nenkova, and McKeown, 2002; Schiffman, Mani, and Concepcion, 2001) is used for sets of articles that are more loosely related and also for biographical documents.


Columbia Newsblaster - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://www.cs.columbia.edu/nlp/newsblaster/summaries/09_10_02_8.html
Go
Internet

Columbia Newsblaster

U.S. Cites New Evidence Saddam Seeking Nuclear Bomb

Summary:

Vice President Dick Cheney said it would be virtually impossible to assemble an international coalition against Iraq if President Saddam Hussein got hold of a nuclear weapon. Iraqi dictator Saddam Hussein is experimenting with unmanned aerial drones to attack the United States or its allies with deadly chemical or biological weapons, Secretary of State Colin Powell charged yesterday. Members of the Bush administration differ about what action to take against Iraq's Saddam Hussein, and no decision has been made on a military strike, Secretary of State Colin Powell said in an interview to be broadcast Sunday. Another exchange between David Frost and Powell, shown by the BBC last week, appeared to highlight sharp differences within the Bush administration over how to deal with the Iraq crisis.

Source Articles:

- U.S. Cites New Evidence Saddam Seeking Nuclear Bomb (Lycos 09/08/2002)
- Cheney Defends Pre-Emptive Doctrine (Lycos 09/08/2002)
- Hard to Build Coalition if Iraq Gets Nuke - Cheney (Lycos 09/09/2002)
- Iraqi dictator Saddam Hussein is experimenting with unmanned aerial dr... (NY Post 09/09/2002)
- Powell: U.S. Iraq Strategy Still Uncertain (FOX News 09/07/2002)

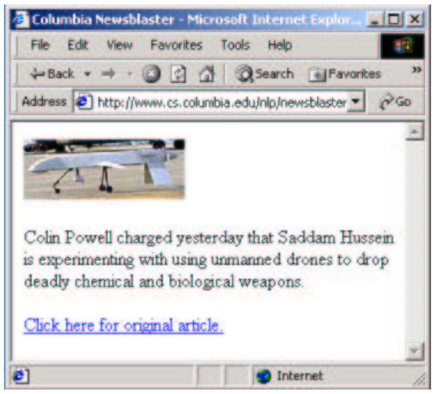
Figure 9.2: When a user clicks on a cluster, an automatically generated summary is displayed, along with links to the original articles and appropriate images found.

Figure 9.2 shows the page with the automatic summary generated for the cluster titled "U.S. Cites New Evidence Saddam Seeking Nuclear Bomb" on September 10, 2002 (you can see the link for this cluster in the Figure 9.1). As you can see, this cluster consists of five articles, and these articles contribute to the automatic summary.[4] One appropriate image has been found in these articles, and it

[4]There is currently work underway to include links from the individual sentences in the summary to the articles that contribute to these sentences.

is displayed at the top of the summary. Below the summary there are links to the original articles that are contained in the cluster.

Appropriate images are found using a set of manually coded rules that I have determined manually by examining many articles from various news sites. The rules are designed to extract most of the appropriate images without also finding inappropriate images such as advertisements. I weight precision higher than recall, since users will not notice if an appropriate image is missing, but inappropriate images that are present will be visible. Some patterns that I have noticed which have led to the rules used to locate appropriate images are: (1) Images are almost always in the same cell as the article or an embedded cell. (2) Images that are jpeg's tend to be appropriate, but other formats are more likely advertisements or logos. (3) Images with a word such as "ad" or "advertisement" in the URL are probably not appropriate. By combining such rules, Newsblaster seems to achieve nearly perfect precision while still recalling the high majority of appropriate images. The discovered images, unlike articles, are not actually downloaded to the Newsblaster server (this would require huge amounts of space, since images are large, and Newsblaster finds thousands a day). Instead, the links to the images are recorded along with the articles in which they are contained.

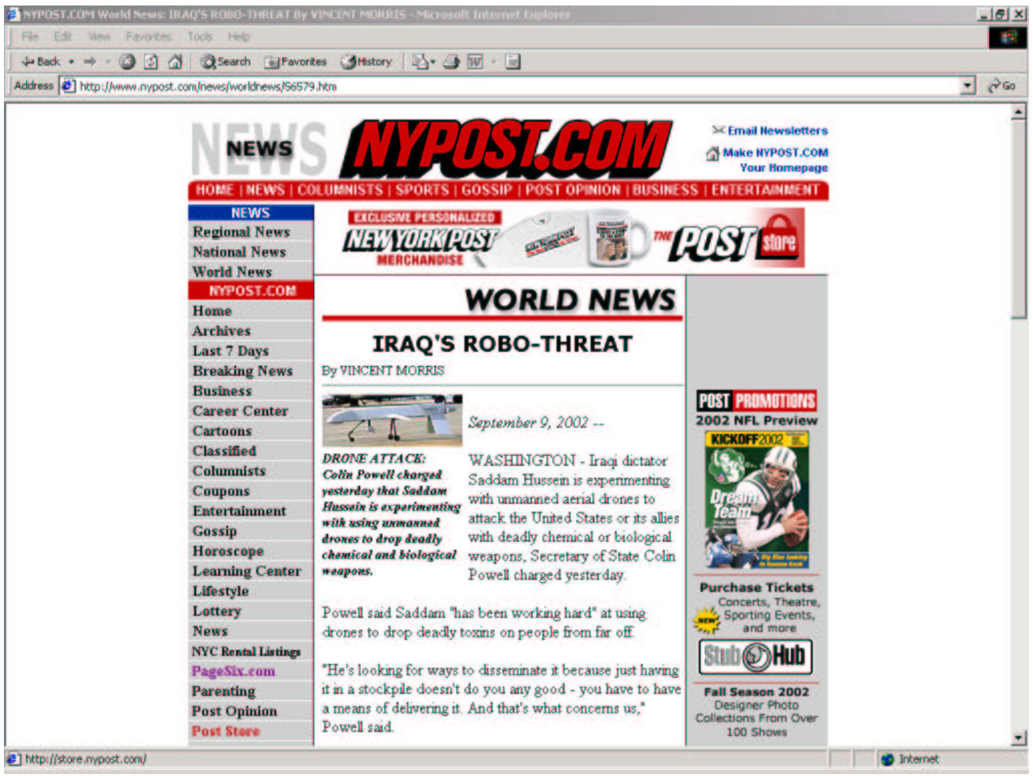

Columbia Newsblaster - Microsoft Internet Explor...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.cs.columbia.edu/nlp/newsblaster
Go
Internet

Colin Powell charged yesterday that Saddam Hussein is experimenting with using unmanned drones to drop deadly chemical and biological weapons.

Click here for original article.

Figure 9.3: When a user clicks on an image at the top of a summary, the full-size image is presented along with its caption, if any, and a link to the original article that it came from.

When a user clicks on an image, he is taken to a new page that displays the full-size image and its caption, if any has been found. Image captions are determined by another set of manually coded rules that I have created. The rules to find appropriate images are relatively simple, but the rules to find captions turn out to be much more complicated, consisting of a large chain of complex regular expressions. Figure 9.3 shows the result of clicking on the image displayed at the top of the summary presented in the Figure 9.2. In this case, you can see that an appropriate caption has indeed been found for the image.


NYPOST.COM World News: IRAQ'S ROBO-THREAT By VINCENT MORRIS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://www.nypost.com/news/worldnews/56579.htm
Go
NEWS
NYPOST.COM
Email Newsletters
Make NYPOST.COM Your Homepage
HOME | NEWS | COLUMNISTS | SPORTS | GOSSIP | POST OPINION | BUSINESS | ENTERTAINMENT
NEWS
Regional News
National News
World News
NYPOST.COM
Home
Archives
Last 7 Days
Breaking News
Business
Career Center
Cartoons
Classified
Columnists
Coupons
Entertainment
Gossip
Horoscope
Learning Center
Lifestyle
Lottery
News
NYC Rental Listings
PageSix.com
Parenting
Post Opinion
Post Store
EXCLUSIVE PERSONALIZED NEW YORK POST MERCHANDISE
THE POST store
POST PROMOTIONS
2002 NFL Preview
KICKOFF 2002
Dream Team
Purchase Tickets
Concerts, Theatre, Sporting Events, and more
NEW
StubHub
Fall Season 2002
Designer Photo Collections From Over 100 Shows
http://store.nypost.com/
Internet

WORLD NEWS

IRAQ'S ROBO-THREAT

By VINCENT MORRIS

September 9, 2002 --

DRONE ATTACK: Colin Powell charged yesterday that Saddam Hussein is experimenting with using unmanned drones to drop deadly chemical and biological weapons.

WASHINGTON - Iraqi dictator Saddam Hussein is experimenting with unmanned aerial drones to attack the United States or its allies with deadly chemical or biological weapons, Secretary of State Colin Powell charged yesterday.

Powell said Saddam "has been working hard" at using drones to drop deadly toxins on people from far off.

"He's looking for ways to disseminate it because just having it in a stockpile doesn't do you any good - you have to have a means of delivering it. And that's what concerns us," Powell said.

Figure 9.4: Clicking on the link below and image and caption brings the user to the original source of the image/caption pair.

Clicking on the link below the image and caption brings the user to the original article that the image and caption comes from. This image happens to come from an article posted at NYPost.Com. Figure 9.4 shows the original article that contains the image and caption shown in the Figure 9.3. The user also could have been taken to this page by clicking on the fourth link under the summary (shown in Figure 9.2). You can see that many other images were also present on the page, but none would have been appropriate to display with the summary. You can also see that there is text in between the image and the caption, but the regular expressions correctly realize that the longer text is the actual caption.

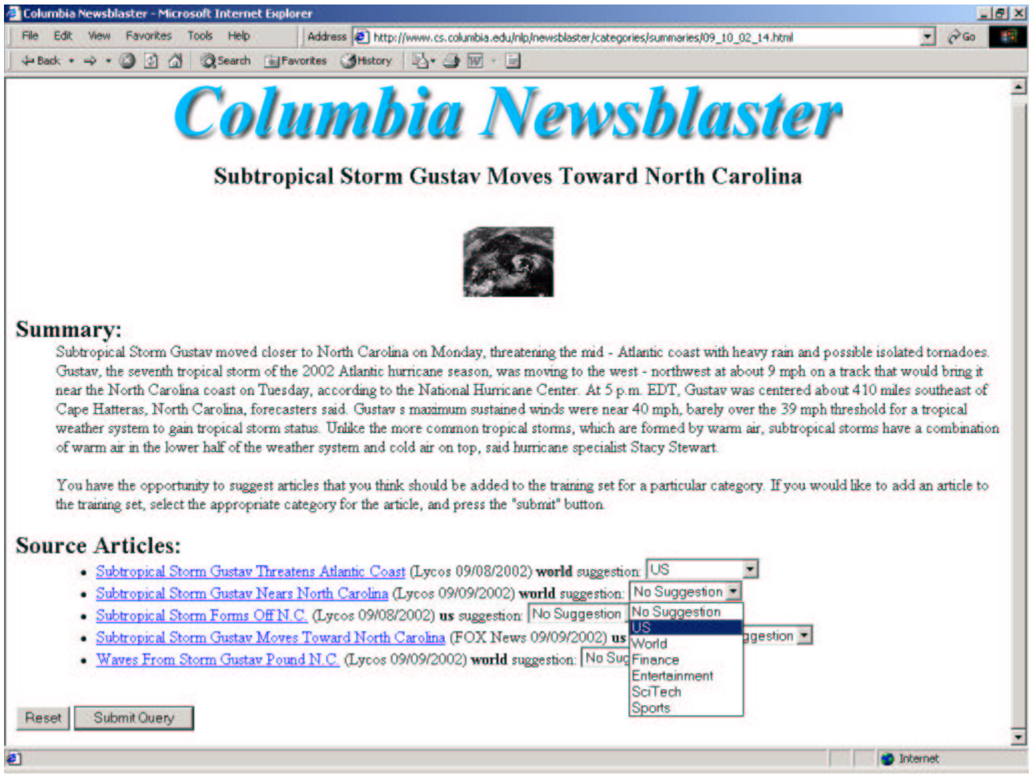
9.2 Improving Newsblaster Categorization

When we first added categorization into useful sections to the Newsblaster interface, there was an attempt to automatically create a training set by using training examples for each category from sources that are extremely likely to fit the categories (without the need of checking). It is explained in Appendix P why this does not work as well as using a more representing training set consisting of manually labeled articles. Ever since the manual training set was created, the high majority of superclusters have been categorized correctly. Today, there as often as many as 50 superclusters in a given day, and usually at most two or three are incorrectly classified.

Still, it happens, especially when a new topic occurs that is unlike anything that has been seen before. One noteworthy example is the past Olympic games. Many of the stories concerning the Olympics clearly belonged in the *Sports* section, as all humans who I asked agreed. However, the sports that occur as part of the Olympics are often sports that do not occur elsewhere, and all the talk of various foreign countries fooled our classifier such that it often placed the stories in the *World News* section instead. To allow categorization to improve over time, I have therefore implemented an alternative version of Newsblaster, also built daily, which allows users to suggest categories (sections) for individual articles that they believe have been labeled incorrectly. If the Newsblaster team agrees with the suggested category, the article is added to the training set, so that similar mistakes are less likely to happen in the future. (This alternative interface also allows users to suggest adding correctly labeled articles to the training set, which is sometimes helpful if similar articles are labeled wrong.) This alternative interface is currently available at http://www.cs.columbia.edu/nlp/newsblaster/categories.

Looking back at Figure 9.1, you may notice that of the five superclusters (and eight clusters) visible, all but the last clearly belong in the *World News* section; however, the last of these superclusters has been classified incorrectly, as it concerns a tropical storm heading towards North Carolina. The focus of all five articles in the cluster is about the threat to this U.S. state; however, three of the articles have been misclassified as belonging to the *World News* category. Why this has happened is unclear; perhaps articles about tropical storms are more common in this section, or perhaps the name "Gustav" has thrown off the system. In any case, it might be helpful to add some of these examples to the training set as examples of the *U.S. News* category so that mistakes like this are less likely to happen again.

The first page of the alternative interface is exactly like the regular interface, so it would look just like what you see displayed in Figure 9.1 (except that the URL would be different). However, when you click on a cluster to see the summary and links to the original source articles, this version Newsblaster also tells you how each article has been labeled, and drop-down boxes are provided that allow the users to suggest alternatives. Figure 9.5 shows the sample summary page for the incorrectly labeled cluster when using the alternative version of Newsblaster just discussed. As you can see, next to each link to an original article, the automatic label that was assigned to the article is specified, and a drop-down box allows the user to make a new suggestion (or confirm the original one, if the user wants to add a correctly classified article to the training set). In this case, two of the articles are correctly classified as *U.S. News*, but three are misclassified as *World News*. The


Columbia Newsblaster - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.cs.columbia.edu/nlp/newsblaster/categories/summaries/09_10_02_14.html
Columbia Newsblaster

Subtropical Storm Gustav Moves Toward North Carolina

Summary:

Subtropical Storm Gustav moved closer to North Carolina on Monday, threatening the mid - Atlantic coast with heavy rain and possible isolated tornadoes. Gustav, the seventh tropical storm of the 2002 Atlantic hurricane season, was moving to the west - northwest at about 9 mph on a track that would bring it near the North Carolina coast on Tuesday, according to the National Hurricane Center. At 5 p.m. EDT, Gustav was centered about 410 miles southeast of Cape Hatteras, North Carolina, forecasters said. Gustav s maximum sustained winds were near 40 mph, barely over the 39 mph threshold for a tropical weather system to gain tropical storm status. Unlike the more common tropical storms, which are formed by warm air, subtropical storms have a combination of warm air in the lower half of the weather system and cold air on top, said hurricane specialist Stacy Stewart.

You have the opportunity to suggest articles that you think should be added to the training set for a particular category. If you would like to add an article to the training set, select the appropriate category for the article, and press the "submit" button.

Source Articles:

- Subtropical Storm Gustav Threatens Atlantic Coast (Lycos 09/08/2002) **world** suggestion: US
- Subtropical Storm Gustav Nears North Carolina (Lycos 09/09/2002) **world** suggestion: No Suggestion
- Subtropical Storm Forms Off N.C. (Lycos 09/08/2002) **us** suggestion: No Suggestion
- Subtropical Storm Gustav Moves Toward North Carolina (FOX News 09/09/2002) **us**
- Waves From Storm Gustav Pound N.C. (Lycos 09/09/2002) **world** suggestion: No Sug

Figure 9.5: An alternative version of Newsblaster allows users to suggest categories for articles they think have been labeled incorrectly so that similar mistakes are less likely in the future.

state of the page as shown in the figure is such that the user has already suggested a correction for one of the misclassified articles and is in the process of suggesting a second correction. Once the user is finished specifying labels for whichever articles he or she chooses, the user can click on the "Submit Query" button to submit the suggestion.

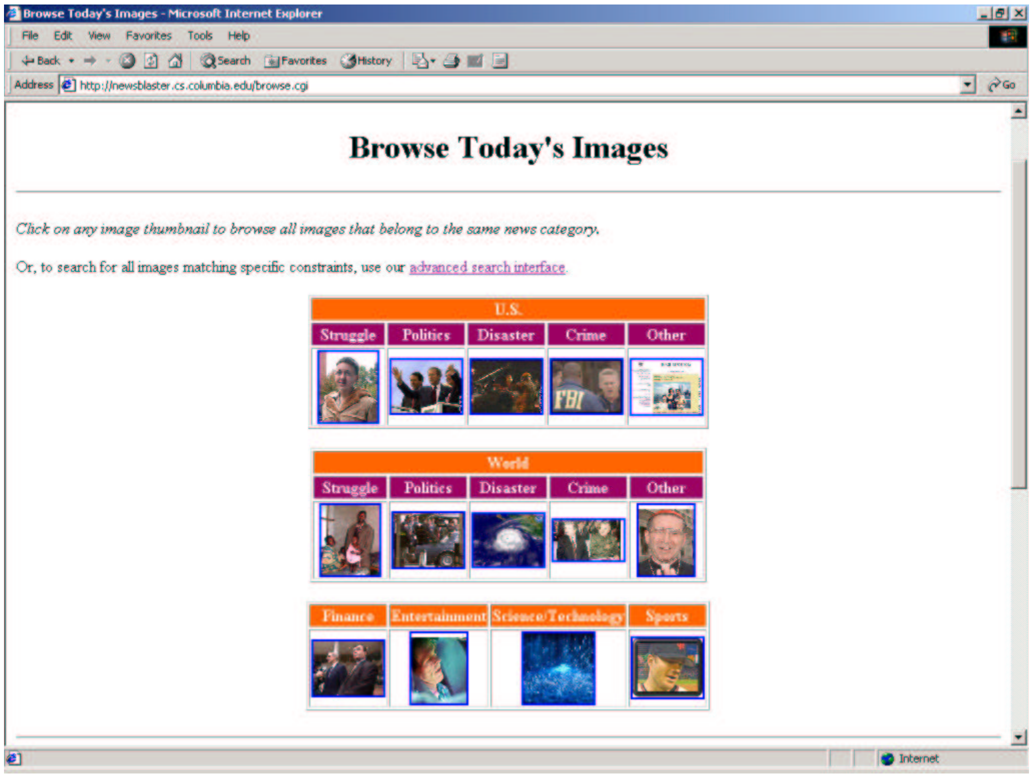
Submitting suggestions does not automatically add the articles to the training set. Rather, articles are copied to a temporary location, and all suggestions are recorded in a single file. Another script exists that implements a simple text interface that allows anyone with access (currently, the NLP group at Columbia University) to review the suggestions (by displaying each article with its suggested category). This person can then either choose to add the article to the training set, delete the article without adding it to the training set, or postpone the decision until later. Once all additions have been made, another script exists that takes care of retraining BINS based on the new training set (and then the new trained model needs to be checked in so it can propagate to the live version of Newsblaster).

9.3 Browsing Images

Note the link at the top of Figure 9.1 reading "New! Browse all of today's images!". When a user clicks on this link, he or she is taken to Newsblaster's image browsing interface, one of my personal contributions to the system. This interface allows users to access all of the news-related images found during Newsblaster's crawling phase. Many of these images do not show up in any summary, because only articles that wind up in a cluster with at least some specified minimum number of articles (currently four) make it to the main Newsblaster interface. There are typically thousands of images found each day, whereas only hundreds are associated with a summary. The image browser interface allows users to browse all images belonging to a given category (or combination of categories), or to search for images with specific properties.

Figure 9.6 shows the first page of Newsblaster's image browsing interface from October 22, 2002. Images are classified into the Newsblaster categories (described in Section 9.1.3), and those images that fall into the *U.S. News* or *World News* categories are subcategorized into the Events categories (first described in Section 3.1.2.3). A single image is then randomly selected to represent each of the categories or category combinations, and these images are displayed as shown in Figure 9.6. Each time the page is refreshed, a new representative image is chosen to fill each slot.

Figure 9.7 shows the screen that results if the user clicks on the image representing the *U.S./Politics* category combination as shown in Figure 9.6. This screen

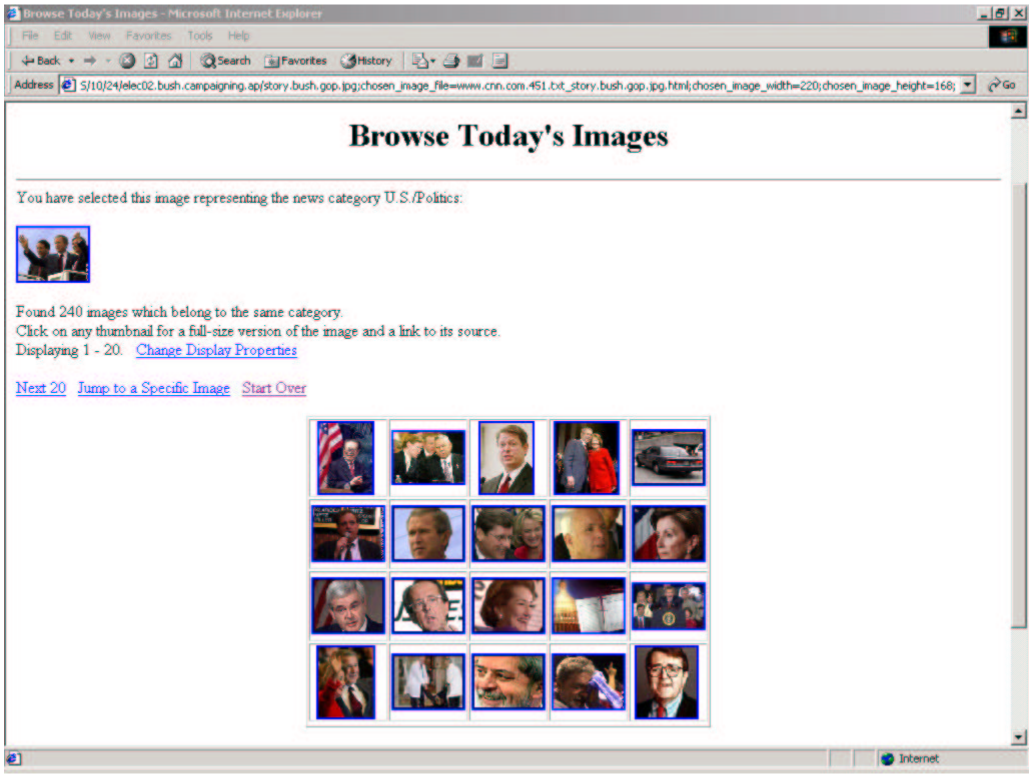

Browse Today's Images

Click on any image thumbnail to browse all images that belong to the same news category.

Or, to search for all images matching specific constraints, use our advanced search interface
Browse Today's Images - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://newsblaster.cs.columbia.edu/browse.cgi
Go
U.S.
Struggle
Politics
Disaster
Crime
Other
World
Struggle
Politics
Disaster
Crime
Other
Finance
Entertainment
Science/Technology
Sports
Internet

Figure 9.6: Newsblaster's image browsing interface displays a randomly selected representative image for each of the major categories (or category combinations).

displays thumbnails for the first set of all images that belong to the same category combination. By default, the image thumbnails are displayed 20 at a time in rows of 5 with each image placed in a square occupying a width and height of 70 pixels (but the image keeps its original proportion, so there might be white space above and below or to the sides of an image). The user can jump forward or backwards one screen at a time, or he/she can decide to jump to any specific image. Assuming the user arrived at this screen by clicking on an image from the grid presented at the top of the browsing interface (an alternate means of getting to this screen is discussed shortly), the image the user clicked on is also displayed at the top of the new grid; this way, if the user is specifically interested in this image, he or she does not have to search for it, which might be annoying if many images share the


Browse Today's Images - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address S/10/24/elec02.bush.campaigning.ap/story.bush.gop.jpg;chosen_image_file=www.cnn.com.451.txt_story.bush.gop.jpg.html;chosen_image_width=220;chosen_image_height=168;
Go
Internet

Browse Today's Images

You have selected this image representing the news category U.S./Politics:

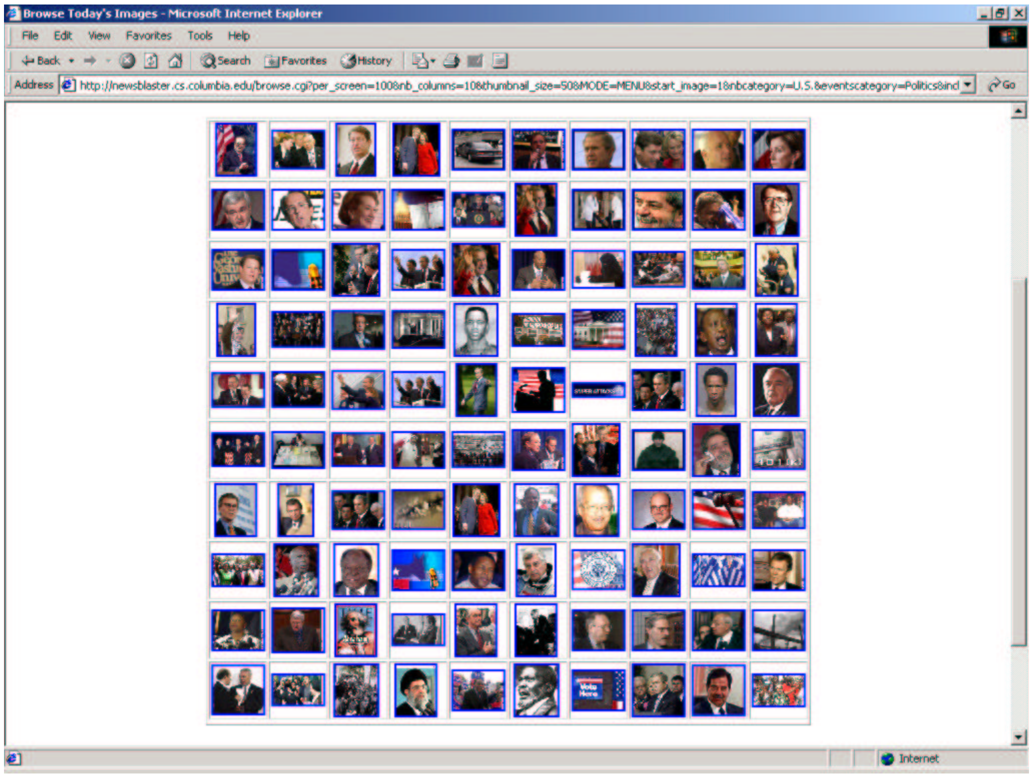
Found 240 images which belong to the same category.
Click on any thumbnail for a full-size version of the image and a link to its source.
Displaying 1 - 20. Change Display Properties

Next 20 Jump to a Specific Image Start Over

Figure 9.7: After a category is selected, Newsblaster allows the user to browse all images with the same category (or category combination).

current category. (Originally, this functionality was not part of the image browsing interface, and it was requested by multiple people.) If the user clicks on any image thumbnail, he or she is taken to the page for that image, including a full-size version of the image, the image's caption (if one has been found), and a link to the original article that the image comes from. These pages are the same as those that a user is taken to when he or she clicks on an image at the top of a summary (as shown in Figure 9.3).

The user is free to change the display properties; for example, Figure 9.8 shows what the screen from Figure 9.7 would look like if the user decides to display 100 images per screen in rows of 10 with each image placed in a square occupying a width and height of 50 pixels. Users with quick Internet access and a large screen

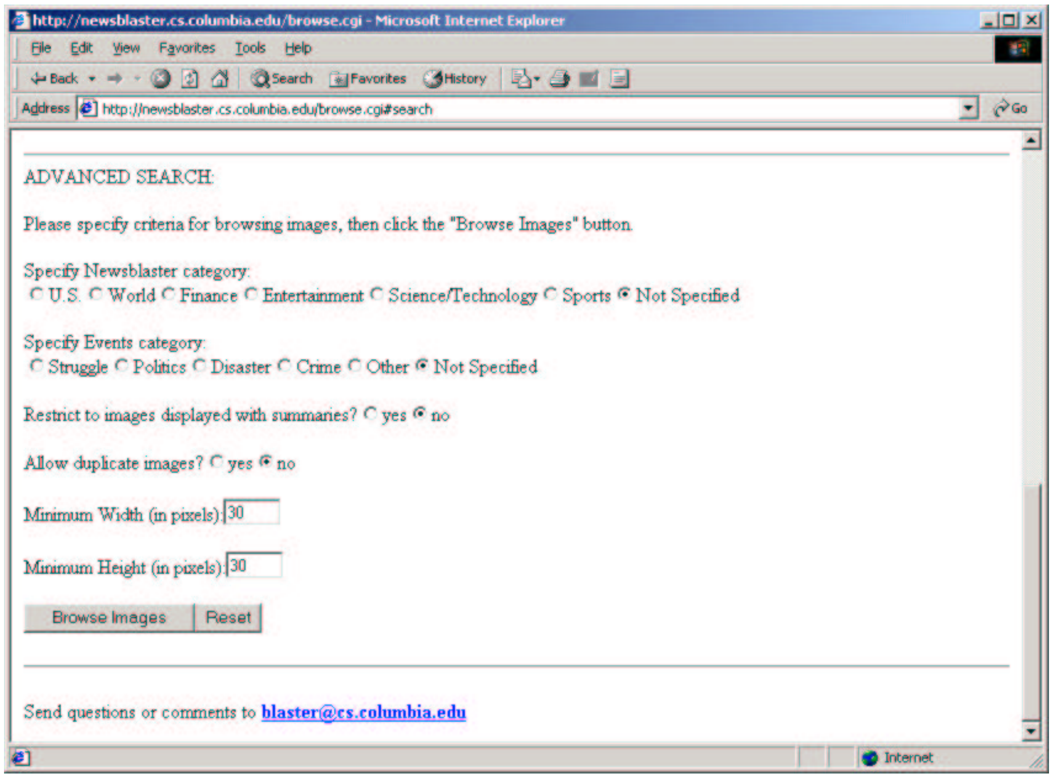

Browse Today's Images - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address
Go
Internet

http://newsblaster.cs.columbia.edu/browse.cgi?per_screen=100&nb_columns=10&thumbnail_size=50&MODE=MENU&start_image=1&nbcategory=U.S.&eventscategory=Politics&ind

Figure 9.8: The image browsing interface allows a user to change the display properties while browsing images.

may choose to do this.

Notice at the top of Figure 9.6 that there is a link to an "advanced search interface". This interface exists below the grid of representative images, and it is shown in Figure 9.9. This interface allows the user to specify the specific properties he or she is interested in, and all such images that match these properties are found and presented in the same manner is shown in Figure 9.7. I hope to make this advanced search interface more useful by adding the ability of the user to enter text that Newsblaster will search for in either captions or articles. By allowing the user to search for images in specific categories that have specific associated text, a user should be able to find images that suit his or her interest. I also hope to add the *Indoor* versus *Outdoor* categories to the search interface. Since many (in fact, the



http://newsblaster.cs.columbia.edu/browse.cgi - Microsoft Internet Explorer

Address http://newsblaster.cs.columbia.edu/browse.cgi#search

ADVANCED SEARCH:

Please specify criteria for browsing images, then click the "Browse Images" button.

Specify Newsblaster category:
U.S. World Finance Entertainment Science/Technology Sports [✓ Not Specified]

Specify Events category:
Struggle Politics Disaster Crime Other [✓ Not Specified]

Restrict to images displayed with summaries? yes [✓ no]

Allow duplicate images? yes [✓ no]

Minimum Width (in pixels): 30

Minimum Height (in pixels): 30

Send questions or comments to blaster@cs.columbia.edu

Figure 9.9: Newsblaster also allows users to search for images with specific properties.

majority) of images found by Newsblaster do not have associated captions, and full articles generally do not do a good job for this set of categories (see Appendix G), low-level image features will probably have to be used to classify many of the images, as described in Chapter 8. When a caption is present, a combination of text and image features can be used.

9.4 Research Related to Newsblaster

The automatic tracking and summarization of news from the web is a recent endeavor, and Newsblaster is surely one of the first, if not the first, major efforts in this area. The most similar research project of which I am currently aware is the NewsInEssence project led by Dragomir Radev at the University of Michigan

(Radev et al., 2001). Like Newsblaster, NewsInEssence locates news articles on the web, clusters related articles, and summarizes each cluster. NewsInEssence does not categorize clusters into sections, nor does it locate images contained in articles (my two main personal contributions to Newsblaster). NewsInEssence can be accessed on the web at http://www.newsinessence.com.

Very recently, Google has created their own similar project which they call Google News, and this can be accessed on the web at http://news.google.com. Currently testing a BETA version, Google News clusters and categorizes news, using, in fact, the the same categories as Newsblaster with the addition of a *Health* category. Presentations of clusters include links to the related articles and a single picture extracted from one article of each cluster. The beginning of each article is shown next to the link to the article, and there is no further attempt to summarize. A FAQ about the system is available at http://news.google.com/help/about_news_search.html, which presents information at a very non-technical level. I have written to the staff requesting more information, but their response, which took 17 days, was minimal and non-informative.

Newsblaster takes advantage of much of the research that has previously been conducted by members of Columbia's Natural Language Processing group. This includes research on clustering (Hatzivassiloglou, Gravano, and Maganti, 2000) and topics related to multi-document summarization (Barzilay, Elhadad, and McKeown, 2002; Schiffman, Nenkova, and McKeown, 2002; Hatzivassiloglou et al., 2001; McKeown et al., 2001; Schiffman, Mani, and Concepcion, 2001; McKeown et al., 1999; Barzilay, McKeown, and Elhadad, 1999). The categorization performed by the system uses my BINS system which has been described in Chapter 5, and an older version of the system is also described in (Sable and Church, 2001). Links to these academic papers related to Newsblaster can be found at http://www.cs.columbia.edu/nlp/newsblaster/papers.

Work on image browsing interfaces includes Bederson's PhotoMesa (Bederson, 2001), which displays user's personal, digital photographs based on how they are stored in directory structures. PhotoMesa can display many images from multiple directories on a single screen, and it zooms to regions of interest determined by a user's mouse movements. This is nice, and I personally know people who use PhotoMesa to browse their own pictures, but it is beyond the scope of my research which does not focus on user interfaces.

Work on image search interfaces includes the work led by Marti Hearst on Flamenco (Hearst et al., 2002). Flamenco takes advantage of *hierarchical-faceted metadata* to aid users to find images from a very large collection. The interface uses the metadata in such a way as to allow users to go back and fourth between following links and refining textual queries until they find what they are looking for.

9.5 Concluding Discussion of Newsblaster

Newsblaster helps add credence (not that I think this is necessary) to the research I've been working on for the past five years. It demonstrates how various achievements in the field of Natural Language Processing can come together to create a useful and interesting system. In Section 2.2 of my thesis, I list some of the potential pragmatic uses of text categorization, and one of them is the classification of news into topical sections for browsing purposes. The core of my thesis deals with images, and I have repeatedly shown that text categorization techniques can be applied to text associated with images to label the images in order to aid browsing or search capabilities. Newsblaster takes advantage of my work, combining it with the work of other researchers who have been working in the areas of clustering and summarization. The result of this joint effort is a system that has already caught the attention of the public and the press. Companies have expressed interest, as well, in applying the technology driving Newsblaster to their own data. The fact that Newsblaster automatically updates itself every day (and will soon update itself incrementally throughout the day) tests our work constantly; although we can't actually evaluate the results of these daily tests in a formal sense, a perusal of Newsblaster on a typical day assures me that all components are working well. I personally enjoy using Newsblaster as a major source of news.

Chapter 10

Conclusions

10.1 A Recap of the Two Paradigms of Research

Research in the text categorization literature almost always fits into at least one of two paradigms. The more common of these paradigms concerns the exploration of new machine learning techniques to improve results. Some of the research discussed in this thesis fits into this paradigm, including the use of density estimation discussed in Chapter 4, the use of bins discussed in Chapter 5, and the combination of high-precision, low-recall rules with other systems discussed in Chapter 7. The other paradigm concerns the use of novel representations of documents. My work discussed in Chapter 6, involving the use of deeper linguistic processing to represent an image using only the main subject and verb of its caption - as opposed to a typical bag of words approach used by almost all other text categorization systems - fits into this paradigm. The work discussed in Chapter 8, involving the use of low-level image features such as color histograms to represent documents, clearly fits into this paradigm as well. Most of the text categorization literature involves research that fits only into the first paradigm, as you can see from the survey of the field I provide in Chapter 2 (exceptions are discussed in Section 2.4.3). Because

I have been dealing with images, however, the properties of text and categories have varied from the norm, leading to the NLP based work, while the availability of image features has led to the image feature based work.

Machine learning techniques are general, and once a system relying on such a technique has been developed, it is usually relatively simple to move to a new task. For example, to apply an existing text categorization system to a new set of categories, the most expensive step is usually to collect training examples for each of the new categories. Research involving novel representations for documents is often more specific. My work involving low-level image features, for example, clearly only applies to images. In the literature, one of the most successful uses of novel representation involves the use of hyperlinks for retrieval from the World Wide Web, and this clearly only applies to web pages. Although it is certainly possible that advanced NLP might eventually prove to useful for text categorization in general, I make no claims that the specific approach I use is general; I have been successful using deeper linguistic processing for a specific task involving the focus of images. While research that fits into this paradigm tends to be more specific, such research is often very interesting, and sometimes it can lead to substantial improvement over standard approaches.

10.2 A Summary of My Main Contributions

Two of the contributions of this thesis are general, and do not clearly fall into just one of the two paradigms just described in Section 10.1. These contributions are:

- *The exploration of the use of text to categorize images.* This represent the core of my dissertation, and all of the other contributions relate to this one in some way. The categorization of images is important for many of the same reasons as text categorization in general. The increasing availability of free-floating images on the Web, the creation of large corpora of images, and the commonality of personal collections of digital photographs (some of which have annotations) all lead to the necessity of better ways to automatically label images to aid tasks such as browsing, filtering, and searching. In order to have the means of conducting the appropriate research, *I have created a multimedia corpus consisting of news documents with embedded captioned images and multiple data sets representing various levels of abstraction.* I hope that this corpus becomes popular in the field so that researchers conducting similar research can compare their results to mine. Almost all of the existing text categorization literature involves the categorization of text-only documents, and usually involves experimentation with one of a few publicly available corpora (a list of such corpora is provided in Section 2.8). I believe that my corpus can serve as an important resource, since the properties of the text and the categories associated with images are often quite different from those associated with typical full-length textual documents such as articles, e-mails, and web pages, and these varying properties affect which approaches perform best.

- *The categorization of news and an image browsing interface for Newsblaster.* Newsblaster, which has already captured the attention of the public and press, is a publicly accessible system that showcases my work and also the work of many other students within the NLP group at Columbia. The Newsblaster system extracts, clusters, categorizes, and summarizes news and related images, and it provide a user-friendly interface for anyone with access to the web to browse and read the news or to browse and search for news related images. Newsblaster is automatically updated every day, and soon will be updated throughout each day. My personal contributions include the categorization of stories into sections that are typical of a newspaper or manually created news site, the inclusion of images that are displayed with summaries, and a user-friendly image browser that allows users to browse the current day's images or search for images that match various properties. This contribution does not really involve new research; rather, it is an implementation contribution involving the integration of research from many students that has culminated in a pragmatic and popular system.

The remaining contributions of my thesis each fall into one of the two paradigms described in Section 10.1. The contributions that fit into the first paradigm, i.e. those concerning the use of novel machine learning techniques to improve performance, are:

- *The introduction of two novel machine learning approaches towards text categorization involving the use of density estimation and bins.* Density estimation is a statistical technique used to estimate a probabilistic confidence measure for each of a system's predictions, and it often also improves accuracy. As explained in Sections 8.3.3 and 10.3, I believe that the confidence measures provided by density estimation can be used to intelligently combine systems together, although I am leaving this for future work. The use of bins provides a mechanism for empirically estimating accurate term weights for words with scarce evidence, and for determining which statistical features of a word are important. Both of these systems are competitive with other advanced approaches, and my BINS system performs especially well for two of the data sets of my own corpus. (With the appropriate settings combining bin-based weights and Naive Bayes weights as explained in Section 5.7, BINS beats all other systems tested for the *Indoor* versus *Outdoor* data set, first described in Section 3.1.2.2, and the Events data set, first described in Section 3.1.2.3.) The BINS system provides a user-friendly interface that allows for the use of many optional features (some are discussed in Appendices I, J, K, L, and M), and it also allows for the use of unlabeled data (discussed in Appendix N). I am ready to make the system public, and I hope that researchers use it for their own data sets.

- *The use of high-precision, low-recall rules to improve the results of other systems.* I have shown that the use of high-precision, low-recall rules (i.e. rules that are individually rarely applicable but very accurate when they apply) can be used to improve the performance of standard systems for tough categorization tasks. I have demonstrated that by relying on such rules when possible, and falling back to some standard system otherwise, the accuracy of the standard system almost always increases, since the rules are more likely to be correct for the cases to which they apply (at least for certain difficult categorization tasks).

Although they are much less common in the text categorization literature, my thesis also offers contributions that fall into the second paradigm discussed in Section 10.1, i.e. research involving novel representation of documents. These contributions are:

- *The integration of NLP techniques and traditional IR techniques for categorization.* As I have explained in Section 6.7, the information retrieval literature shows a lot of interest in combining NLP and IR, but the results have been mixed, at best. I have clearly shown, however, that for specific categories referring to the focus of an image, deeper linguistic processing is necessary for optimal performance. I have demonstrated this first with experiments involving human volunteers who have predicted categories for images after viewing text under varying conditions, and then by implementing a system relying on parsing and a novel measure of word-to-word similarity that outperforms all standard systems tested for the same task by a considerable margin. Tasks such as these defy standard text categorization systems because not all words associated with each document are important, and some are misleading. To determine the important words, more advanced NLP techniques are therefore required.

- *Combining text and image features for the categorization of images.* I have demonstrated one technique to combine text and low-level image features for the categorization of images. Clearly, this is only applicable when dealing with images. The use of image features may be necessary when text is not available, and when both are available, a combination of the two is likely to outperform (although perhaps by a small margin) the better of the two individual approaches. One reason that combining text and image features may be more promising than combining two approaches that both use text is that there is no reason to believe that techniques relying on entirely different features will perform well for the same documents. By relying on confidence measures for the predictions of the text based system and the image based system, an approach combining the two can choose the prediction that is more likely to be accurate. My work in this area so far has used only one low-level image feature (color histograms) and a simple approach of combining text and image together; even now, some improvement has been noticed for all systems with one system experiencing substantial improvement.

10.3 Future Work and Limitations

There are many components of my research that could lead to future work, either for myself or others who choose to explore this area. To start with, in the domain of general text categorization, there is a lot more that can be done with bins. In my concluding section of my chapter on BINS (Section 5.8), I state that there is likely enough potential in bins that research in this area could constitute a thesis in itself, and I certainly still believe this. In addition to being a very competitive approach towards text categorization, binning provides a mechanism for testing which features of a word are important. By training a system on a training set using a variety of features, and examining the resulting model, one can see which features made a large difference for term weights. In other words, if the term weights for a set of bins that differ only in one feature (i.e. every other feature is the same for this particular set of bins) tend to be quite different from each other, than this feature is likely important (i.e. it is indicative of the likelihood of a word showing up in a document of a particular category). Therefore, one can test the importance of features without even testing the system (using the trained model) on a test set. Of course, even if a particular feature seems to be useful, it does not necessarily mean that performance will improve by using the feature. As has been mentioned in Chapter 5 and several of the appendices about optional binning features, adding new features makes bins smaller, and therefore less accurate; performance will only improve if the benefits of more indicative information outweighs the negative effect of potentially less accurate bins.

In particular, I believe that bins might provide an excellent way to incorporate unlabeled data for use with text categorization tasks. Whereas most uses of unlabeled data discussed in the text categorization and IR literature (e.g. see (Yarowksy, 1995), (Blum and Mitchell, 1998), and (Abney, 2002)) involve procedures which iteratively add unlabeled documents with confident predictions to a training set and then treat these documents like any others, the use of bins allows unlabeled documents to be used but weighted less than others by creating a new binning feature. I have discussed my efforts in this area so far in Appendix N, and unfortunately, I have not yet seen positive results. However, this might be because I started with full training sets, and the results of Appendix O suggest that I may have already reached a level of diminishing returns for my own data sets. It is likely that more improvement is certainly possible with enough new data, but unlabeled data is probably never quite as good as manually labeled data, and so the task I set out to accomplish might be difficult. A major question that still remains is whether unlabeled data can be used for my data sets along with a much smaller training set to achieve performance comparable to that resulting from a full training set.

The limitations of bin-based text categorization systems are the same as those for all standard text based categorization systems. In other words, binning is a very good bag of words approach, but it is still a bag of words approach. Smoothing is helpful; I expect that my BINS system will usually beat a Naive Bayes approach and rarely do worse. However, as we have seen in Chapter 6, there are tasks for which linguistic processing in necessary to achieve optimal performance. Even humans who are shown bag of words representations for the first sentences of image captions can not achieve nearly the same performance as humans who see the same words in their original order for the *Disaster* image data set. There is no reason to believe that statistical bag of words systems can perform better than humans, so when tasks such as this one arise, bin based approaches and all bag of words approaches have an upper bound that is lower than an NLP based approach. Even when dealing with tasks for which bag of word approaches perform well, there is no reason to believe that the upper bound is perfect performance. Even for these tasks, there may be some documents for which a bag of words is misleading such that any standard system would not predict the correct category.

Another area that I'm sure leaves room for a lot of important future work is the combination of text and image features for the categorization of images. Although combining systems has had mixed results in the literature (see my discussions in Sections 7.5 and 8.4), these endeavors have involved the combination of systems that all use text and bag of words approaches. Even though the techniques that have been combined are different, the features upon which predictions are being made are the same, and so it is likely that the various systems do well on the same documents. On the other hand, there is no reason to believe that images with very indicative text (i.e. text that provides obvious clues as to the correct category) also have very indicative low-level image features, and vice versa. So the documents that are the hardest to predict for one type of system might not be so difficult for the other type of system.

The combination of text and image features is one of the most recent areas I have worked on, and it is further from my focus that most of research discussed in this dissertation. In Chapter 8, I describe one technique to combine text and image features, and I have implemented this for a particular set of categories. My results so far are promising; one text based system improves substantially, and all improve by at least a small margin. As I explain in Section 8.3.3, I believe that it is possible to do much better. I am only using one low-level image feature, namely color histograms; it is likely that the use of additional low level features will push the image results and the combined results further ahead. Additionally, there are better ways to actually combine the results. Rather than use only the confidence of the image based system to decide which prediction to use, the confidence of both types of system should be used and compared to each other. Density estimation (explained in Chapter 4) can be used to convert predictions of systems to the same scale (probabilities ranging from zero to one); the experiments described in Appendix H provide evidence that these probability estimates are reasonably accurate.

The limitations of an approach that combines text and image features is related to the intersection of limitations of an approach using only text and an approach using only image features. In other words, there are always certain documents for which the text is not enough to predict the correct category, and there are always certain documents for which image features are not enough to predict the correct category. Limitations of text based approaches have already been discussed. Limitations of image-feature systems have been discussed to some extent at the start of Chapter 8. For many tasks, image features are not appropriate for categorization, since low-level image features such as color may not be indicative of high-level topical categories, and the state-of-the-art in object recognition is still not good enough for this purpose. When low-level image features are still useful for a categorization task, there is still no reason to believe that they will be indicative of the correct category for every case. Since text and image features represent two entirely distinct feature sets, it is likely that a combination of the two can do better than either individually; however, if there is any overlap between the sets of images for which the two approaches lead to incorrect predictions, a combination of the two will doubtfully achieve correct predictions for these images that are part of this overlap.

Another area that clearly merits future work is the use of more advanced NLP techniques to categorize images. I feel that the results of Chapter 6, which describes the use of deeper linguistic processing to achieve a performance substantially better than all tested standard systems for categories involving the focus of images, are among the most impressive in the thesis. As recently stated in Section 10.1, research such as this, involving a novel representation for documents (as opposed to a bag of words representation) is usually less general than standard text categorization research. As researchers explore more domains and categories related to images, I believe that tasks such as the one described in Chapter 6, which arose naturally in the course of my research, will continue to appear, and when they do, approaches similar (but not identical) to the one used in that chapter will be useful.

Of course, it is likely that, at times, NLP based approaches may also be useful for categories that do not involve images. However, I think that such approaches may be more feasible when dealing with images, since the text is often in the form of a caption in which a single sentence describes the image, and single sentences are clearly easier to parse and make sense of than full-length text documents. I also believe that the categories associated with images are more likely to rely on focus (involving, for instance, the determination of the primary object in the image and what that thing is doing). When dealing with such categories, it is more likely that not all of the words in the available text are important, and so linguistic processing becomes helpful. As I have explained in Section 6.7, the information retrieval literature has shown a lot of interest in combining NLP and IR, but the results have been quite mixed, at best (Strzalkowski, Lin, and Perez-Carballo, 1998; Strzalkowski, 1999). I do not think it is coincidental that two of the biggest previous successes in this field (the works of Smeaton and Quigley (1996) and Elworthy (2000), which are also discussed in Section 6.7) both pertain to the retrieval of images.

NLP based approaches towards text categorization are limited by the state-of-the-art in other areas of NLP such as parsing and, more generally, natural language understanding. I would like to think that it is technically possible for NLP based systems to perform as well as humans for text categorization tasks. (There is little reason to believe that automated systems will ever do much better than that.) However, this would require near perfect tools, and perhaps better NLP resources as well. For now, what makes it so hard to produce an effective NLP based system is that mistakes are introduced at every level. For example, my system described in Chapter 6 relies first on a tagger, then on a parser, and later on a measure of word-to-word similarity. All of these represent very tough problems for which there are no perfect solutions. The system is still beating standard bag of words approaches for the specific task examined in that chapter because syntax is especially important for that task. However, advanced NLP approaches have rarely led to improvement for text categorization tasks in the literature, and the reason is that they are limited by the tools and resources upon which they rely.

10.4 A Final Word

This thesis has turned out to be longer than I originally imagined, especially when all of the appendices are considered. Rather than end the main body of the dissertation with the previous section, which, when you think about it, is a summary of things left undone and things that can never happen, I would like to reiterate the positive aspects of the work once more. I believe that a lot of ground has been covered: a survey of the text categorization field; the focus on images and the creation of a corpus; novel and competitive approaches that fall into the standard text categorization framework; the use of more advanced NLP techniques to handle tough categories; the combination of highly accurate but rarely applicable rules with fall back systems; the use of image features, alone or in combination with text, to categorize images; the Newsblaster system that demonstrates the benefits of the work; and many additional topics discussed throughout the chapters and appendices. Everything is related in some way to the general topic of categorizing images based on associated text. I hope that this thesis, which I feel provides an in-depth exploration of this important topic, serves as a useful reference for many future researchers.

References

Abney, S. 1997. The scol manual (version 0.1b).

Abney, S. 2002. Bootstrapping. In *Proceedings of the 40th Annual Meeting of the Association of Computational Linguistict (ACL-95)*.

Banko, M. and E. Brill. 2001a. Mitigating the paucity of data problem. In *Proceedings of the Human Language Technology Conference (HLT-01)*.

Banko, M. and E. Brill. 2001b. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association of Computational Linguistict (ACL-01)*.

Barzilay, R., N. Elhadad, and K. R. McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, **17**:35–55.

Barzilay, R., K. McKeown, and M. Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Annual Meeting of the Association of Computational Linguistict (ACL-99)*.

Bederson, B. B. 2001. PhotoMesa: A zoomable image browser using quantum treemaps and bubblemaps. In *Proceedings of the 14th Annual ACM Symposium on User Interfaces Software and Technology (UIST-2001)*.

Belkin, N. J. and W. B. Croft. 1992. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, **35**(12):29–38.

Benitez, A. B. and S.-F. Chang. 2002a. Perceptual knowledge construction from annoated image collections. In *Proceedings of the 2002 International Conference On Multimedia & Expo (ICME-02)*, Lausanne, Switzerland.

Benitez, A. B. and S.-F. Chang. 2002b. Semantic knowledge constructions from annotated image collections. In *Proceedings of the 2002 International Conference On Multimedia & Expo (ICME-02)*, Lausanne, Switzerland.

Bennett, P. 2000. Assessing the calibration of naive bayes' posterior estimates. Technical Report CMU-CS-00-155, Carnegie Mellon University.

Bennett, P. 2002. Using asymmetric distributions to improve classifier probabilities: A comparison of new and standard parametric methods. Technical Report CMU-CS-02-126, Carnegie Mellon University.

Berger, A. L., S. A. Della Pietra, and V. J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, **22**(1):39–71.

Blum, A. and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98)*.

Brin, S. and L. Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia.

Burges, C. J. C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2**(2):121–167.

Chakrabarti, S., B. Dom, R. Agrawal, and P. Raghavan. 1998. Scalable feature selection, classification, and signature generation for organizing large text databases into hierarchical topic taxonomies. *VLDB Journal*, **7**(3):163–178.

Chakrabarti, S., B. Domb, and P. Indyk. 1998. Enhanced hypertext categorization using hyperlinks. In *Proceddings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-98)*.

Church, K. W. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing (ANLP-88)*.

Church, K. W. 2000. Empirical estimates of adaptation: The chance of two noriegas is closer to $p/2$ than p^2. In *Proceedings of the 18th International Conference on Computational Linguistics COLING-2000*.

Cohen, W. W. 1995a. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*.

Cohen, W. W. 1995b. Text categorization and relational learning. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*.

Craven, M., D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigan, and S. Slatter. 1998. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*.

Cristianini, N. and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK.

Deerwester, S., S. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, **41**(6):391–407.

Drucker, H., D. Wu, and V. Vapnik. 1999. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, **10**(5):1048–1054.

Dumais, S. and H. Chen. 2000. Hierarchical classification of web content. In *Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-00).*

Duygulu, P., K. Barnard, J. F. G. Freitas, and D. A. Forsyth. 2002. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the Seventh Annual European Conference on Computer Vision.*

Efron, B. and C. Morris. 1977. Stein's paradox in statistics. *Scientific American*, 236(5):119–127.

Efron, B. and R. Tibshirani. 1993. *An Introduction to the Bootstrap.* Chapman & Hall.

Elworthy, D. 1998. Inductive learning alogorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management (CIKM-00).*

Elworthy, D. 2000. Retrieval from captioned image databases using natural language processing. In *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM-00).*

Fellbaum, C., editor. 1998. *WordNet: An Electronic Lexical Database.* MIT Press.

Fleck, M. M., D. A. Forsyth, and C. Bregler. 1996. Finding naked people. In *Proceedings of the European Conference on Computer Vision*, Berlin, Germany.

Fuhr, N. 1988. Models for retrieval with probabilistic indexing. *Information Processing and Management*, **25**(1):55–72.

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

Hatzivassiloglou, V. 1998. *Automatic acquisition of lexical semantic knowledge from large corpora: The identification of semantically related words, markedness, polarity, and antonymy.* Ph.D. thesis, Columbia University.

Hatzivassiloglou, V., L. Gravano, and A. Maganti. 2000. An investigation of linuistic features and clustering algorithms for topical document clustering. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-00).*

Hatzivassiloglou, V., J. Klavans, M. Holcombe, R. Barzilay, M.-Y. Kan, and K. McKeown. 2001. Simfinder: A flexible clustering tool for summarization. In *Proceedings of the NAACL Workshop on Automatic Summarization.*

Hatzivassiloglou, V. and K. R. McKeown. 1993. Towards the automatic identification of adjectival scales: Clustering adjectives according to meaning. In *Proceedings of the 31st Annual Meeting of the Association of Computational Linguistict (ACL-93).*

Hearst, M., A. Elliott, J. English, R. Sinha, K. Swearingen, and K.-P. Yee. 2002. Finding the flow in web site search. *Communications of the ACM*, **45**(9):42–49.

Hehrotra, S., Y. Rui, M. Ortaga, and T. S. Huang. 1997. Supporting content-based queries over images in MARS. In *Proceedings of the IEEE Conference on Multimedia Computing and Systems*, pages 632–633.

Hersh, W., C. Buckley, T. Leone, and D. Hickam. 1994. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-97).*

Jelinek, F. 1998. *Statistical Methods for Speech Recognition.* The MIT Press, Cambridge, Massachusetts.

Joachims, T. 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML-97).*

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning.*

Katz, S. 1996. Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, **2**(1):15–59.

Kerminen, P. and M. Gabbouj. 1999. Image retrieval based on color matching. In *Proceedings of the Finnish Signal Processing Symposium (FINSIG-99).*

Kleinberg, J. M. 1999. Authoritative sources in a hyperlinked enviroment. *Journal of the ACM*, **46**(5):604–632.

Kohavi, R. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95).*

Koller, D. and M. Sahami. 1997. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning (ICML-97).*

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

Lang, K. 1995. NewsWeeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95).*

Larkey, L. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-96).*

Larkey, L. S. and W. B. Croft. 1996. Combining classifiers in text categorization. In *Proceedings of the 19th International ACM SIGIR Conference on Researce and Development in Information Retrieval (SIGIR-96).*

Lewis, D. 1997. Reuters-21578 text categorization test collection, readme file (version 1.2).

Lewis, D. 1998. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the European Conference on Machine Learning.*

Lewis, D., R. Schapire, J. Callan, and R. Papka. 1996. Training algorithms for linear text classifiers. In *Proceedings of the 19th International ACM SIGIR Conference on Researce and Development in Information Retrieval (SIGIR-96).*

Lewis, D. and H. Schutze. 2002. Text categorization and analysis. In *Technical, Business, and Legal Dimensions of Protecting Children from Pornography on the Internet: Proceedings of a Workshop.* National Academy Press, chapter 2, pages 5–10.

Lewis, D. D. 1992. An evaluation of phrasal of clustered representations on a text categorization task. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-92).*

Lewis, D. D. 1995. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-95)*.

Lewis, D. D. and W. A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-94)*.

Li, Y. H. and A. K. Jain. 1998. Classification of text documents. *The Computer Journal*, **41**(8):537–546.

Lodhi, H., C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, **2**:419–444.

McCallum, A. 1996. Bow: A toolkit for statistical language modeling, text retrieval, classification, and clustering. http://www.cs.cmu.edu/˜mccallum/bow.

McCallum, A. and K. Nigam. 1998. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI/ICML Workshop on Learning for Text Categorization*.

McCallum, A., R. Rosenfeld, T. Mitchell, and A. Y. Ng. 1998. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*.

McKeown, K., R. Barzilay, D. Evans, V. Hatzivassiloglou, B. Schiffman, and S. Teufel. 2001. Columbia multi-document summarization: Approach and evaluation. In *Proceedings of the ACM SIGIR Workshop on Text Summarization (SIGIR-01)*.

McKeown, K., J. Klavans, V. Hatzivassiloglou, R. Barzilay, and E. Eskin. 1999. Towards multidocument summarization by reformulation: Progress and prospects. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99).*

McKeown, K. R., R. Barzilay, D. Evans, V. Hatzivassiloglou, J. Klavans, A. Nenkova, C. Sable, B. Schiffman, and S. Sigelman. 2002. Tracking and summarizing news on a daily basis with columbia's newsblaster. In *Proceedings of the Human Language Technology Conference (HLT-02).*

Mitchell, T. M. 1997. *Machine Learning.* McGraw-Hill, New York.

Mosteller, F. and D. Wallace. 1963. Inference in an authorship problem. *Journal of the American Statistical Association*, **58**:275–309.

Mosteller, F. and D. Wallace. 1964. *Inference and Disputed Authorship: The Federalist.* Addison-Wesley.

Niblack, W., R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Pektovic, P. Yanker, C. Faloutsos, and G. Taubin. 1993. The QBIC project: Querying images by content using color, texture, and shape. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases.*

Nigam, K., J. Lafferty, and A. McCallum. 1999. Using maximum entropy for text classification. In *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering (IJCAI-99).*

Nigam, K., A. McCallum, S. Thrun, and T. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, **39**(2/3):103–134.

Paek, S., C. Sable, V. Hatzivassiloglou, A. Jaimes, B. Schiffman, S.-F. Chang, and K. McKeown. 1999. Integration of visual and text-based approaches for the content labeling and classification of photographs. In *Proceedings of the ACM SIGIR Workshop on Multimedia Indexing and Retrieval (SIGIR-99)*.

Pang, B., L. Lee, and S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*.

Pentland, A., R. Picard, and S. Sclaroff. 1994. Photobook: Tools for content-based manipulation of image databases. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases IV*.

Picard, R. W. and T. P. Minka. 1995. Video texture for annotations. *Journal of Multimedia Systems*, **3**(1):3–14.

Porter, M. F. 1980. An algorithm for suffix stripping. *Program*, **14**(3):130–137.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning*, **1**(1):81–106.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaugmann, San Mateo, CA.

Radecki, T. 1982. Similarity measures for boolean search request formulations. *Journal of the American Society for Information Science*, **33**(1):8–17.

Radev, D. R., S. Blair-Goldensohn, Z. Zhang, and R. S. Raghavan. 2001. Newsinessence: A system for domain-independent, real-time news clustering and multi-document summarization. In *Proceedings of the Human Language Technology Conference (HLT-01)*.

Rasmussen, E. 1992. Clustering algorithms. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, chapter 16, pages 419–442.

Ratnaparkhi, A. 1997. A simple introduction to maximum entropy models for natural language processing. Technical Report IRCS-97-08, The University of Pennsylvania.

Ratnaparkhi, A. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, The University of Pennsylvania.

Resnik, P. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, **11**:95–130.

Richardson, R., A. F. Smeaton, and J. Murphy. 1994. Using WordNet as a knowledge base for measuring semantic similarity between words. Technical Report CA-1294, Dublin City University.

Riloff, E. 1995. Little words can make a big difference for text classification. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-95)*.

Riloff, E. 1996. Using learned extraction patterns for text classification. In S. Wermter, E. Riloff, and G. Scheler, editors, *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*. Springer Verlag, Heidelberg, DE, pages 275–289. Volume 1040 of Lecture Notes in Artificial Intelligence.

Riloff, E. and J. Lorenzen. 1999. Extraction-based text categorization: Generating domain specific role relationships automatically. In T. Strzalkowski, editor, *Natural Language Information Retrieval.* Kluwer Academic Publishers, chapter 7.

Robertson, S. E. and K. S. Jones. 1976. Relevancy weighting of search terms. *Journal of American Society for Information Science*, **27**(3):129–146.

Rocchio, J. 1971. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing.* Prentice Hall, chapter 14, pages 313–323.

Rowe, N. C. and E. J. Guglielmo. 1996. Natural-language retrieval of images based on descriptive captions. *ACM Transactions on Information Systems*, **14**(3):237–267.

Ruiz, M. E. and P. Srinivasan. 1999. Hierarchical neural networks for text categorization. In *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99).*

Sable, C. and K. W. Church. 2001. Using bins to empirically estimate term weights for text categorization. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01).*

Sable, C. and V. Hatzivassiloglou. 1999. Text-based approaches for the categorization of images. In *Proceedings of the Third Annual Conference on Research and Advanced Technology for Digital Libraries.*

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

Sable, C., K. McKeown, and K. W. Church. 2002. Nlp found helpful (at least for one text categorization task). In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*.

Sable, C., K. McKeown, and V. Hatzivassiloglou. 2002. Using density estimation to improve text categorization. Technical Report CUCS-012-02, Columbia University.

Sahami, M., S. Dumais, D. Heckerman, and E. Horvitz. 1998. A bayesian approach to fitlering junk e-mail. In *Proceedings of the AAAI Workshop on Learning for Text Categorization*.

Salton, G. 1989. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley.

Salton, G. and C. Buckley. 1988. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, **24**(5):513–523.

Schapire, R. and Y. Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning*, **39**(2/3):135–168.

Schapire, R. E., Y. Singer, and A. Singhal. 1998. Boosting and rocchio applied to text filtering. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-98)*.

Schiffman, B., I. Mani, and K. J. Concepcion. 2001. Producing biographical summaries: Combining linguistic knowledge with corpus statistics. In *Proceedings of the Tenth Conference of the European Chapter of the Association for Computational Linguistics*.

Schiffman, B., A. Nenkova, and K. McKeown. 2002. Experiments in multidocument summarization. In *Proceedings of the Human Language Technology Conference (HLT-02)*.

Schutze, H., D. A. Hull, and J. O. Pederson. 1995. A comparison of classifiers and document represenations for the routine problem. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-95)*.

Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, **34**(1):1–47.

Silverman, B. W. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, London.

Smadja, F. Z., K. McKeown, and V. Hatzivassiloglou. 1996. Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, **22**(1):1–38.

Smeaton, A. F. 1999. Using NLP or NLP resources for information retrieval tasks. In T. Strzalkowski, editor, *Natural Language Information Retrieval*. Kluwer Academic Publishers, chapter 4.

Smeaton, A. F. and I. Quigley. 1996. Experiments on using semantic distances between words in image caption retrieval. In *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-96)*.

Smith, J. R. and S.-F. Chang. 1995. Automated image retrieval using color and texture. Technical Report 414-95-20, Columbia University.

Smith, J. R. and S.-F. Chang. 1996a. Searching for images and videos on the world-wide web. Technical Report 459-96-25, Columbia University.

Smith, J. R. and S.-F. Chang. 1996b. Tools and techniques for color image retrieval. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases IV.*

Smith, J. R. and S.-F. Chang. 1996c. VisualSEEk: A fully automated content-based image query system. In *Proceedings of the ACM Multimedia Conference*, Boston, Massachusetts.

Smith, J. R. and S.-F. Chang. 1997. Visually searching the Web for content. *IEEE Multimedia*, **4**(3):12–20.

Srihari, R. K. 1995. Automatic indexing and content-based retrieval of captioned images. *IEEE Computer Magazine*, **28**(9):49–56.

Strzalkowski, T., editor. 1999. *Natural Language Information Retrieval.* Kluwer Academic Publishers.

Strzalkowski, T., F. Lin, and J. Perez-Carballo. 1998. Natural language information retrieval: TREC-6 report. In *The Sixth Text Retrieval Conference (TREC-6).* NIST Special Publication 500-240.

Sussna, M. 1993. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of the Second International Conference on Information and Knowledge Management (CIKM-93).*

Szummer, M. and R. W. Picard. 1998. Indoor-outdoor image classification. In *Proceedings of the IEEE Workshop on Content Based Access of Image and Video Databases (CAIVD-98).*

Turney, P. D. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association of Computational Linguistict (ACL-02).*

Umemura, K. and K. W. Church. 2000. Empirical term weighting and expansion frequency. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-00).*

Vailaya, A., M. Figueiredo, A. K. Jain, and H. Zhang. 1999a. Bayesian framework for semantic classification of outdoor vacation images. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases VII.*

Vailaya, A., A. Jain, , and H. J. Zhang. 1999b. On image classification: City images vs. landscapes. *Pattern Recognition*, **31**:1921–1936.

van Rijsbergen, C. J. 1979. *Information Retrieval.* Butterworths, London, 2nd edition.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York.

Voorhees, E. M. 1993. Using WordNet to disambiguate word senses for text retrieval. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-93).*

Wacholder, N., Y. Ravin, and M. Choi. 1997. Disambiguation of proper names in text. In *Proceedings of the Fifth ACL Conference on Applied Natural Language Processing (ANLP-97).*

Wang, J. Z., J. Li, G. Wiederhold, and O. Firschein. 1997. Classifying objectionable websites based on image content. In R. Steinmetz and L. C. Wolf, editors, *Proceedings of the Fourth International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS-97).* Springer Verlag, pages 20–30. Volume 1309 of Lecture Notes in Computer Science.

Wayne, C. L. 2000. Multilingual topic detection and tracking: Successful research enabled by corpora and evaluation. In *Proceedings of the Second International Language Resource and Evaluation Conference (LREC-2000).*

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

Wolpert, D. H. and W. G. Macready. 1995. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, The Santa Fe Institute.

Wolpert, D. H. and W. G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, **1**(1):67–82.

Yang, Y. 1997. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, **1**(1/2):67–88.

Yang, Y. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*, **1**(1/2):67–88.

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

Yang, Y. and X. Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99).*

Yarowksy, D. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33th Annual Meeting of the Association of Computational Linguistict (ACL-95).*

Zhu, H. and R. Rohwer. 1996. No free lunch for cross validation. *Neural Computation*, /bf 8(7):1421–1426.

Appendix A

Sample Document

The following is a document from my corpus. The document has been manually labeled as a Disaster document, and the image has been labeled with the categories *Outdoor* and *Workers Responding*. I am showing the full article followed by the image and corresponding caption as they appear in the corpus. For my actual experiments, however, all header information, such as locations and dates appearing at the start of articles or captions, are automatically stripped and not used.

> BRASILIA, Brazil (Reuters) - Dry winds fanned fires deeper into the rainforests of Brazil's northern Amazon Monday as authorities deployed more equipment and manpower in a bid to tackle the blazes.
>
> About 30 Brazilian army jungle communications specialists were sent to remote Roraima state on the border with Venezuela as two more Argentine water-carrying helicopters were preparing to join the firefighting effort, bringing the total number of helicopters to four.
>
> But Roraima governor Neudo Campos said the firefighting operation – which also involves Venezuela – was still too small and warned that the situation might deteriorate.
>
> "The situation is extremely serious and it has all the elements to

turn into a new Indonesia,” Roraima state governor Neudo Campos said.

Fires ravaged large areas of forest in Indonesia last year, casting choking smog over much of southeast Asia. New outbreaks have been reported this week.

The Brazilian forest fire – the worst in recent memory – began in January when subsistence farmers ignored government warnings not to use ’slash and burn’ tactics to clear their land and watched helplessly as the flames spread quickly over the savannas.

Amid one of the region’s worst droughts, blamed on the El Nino weather phenomenon, the fires are now eating into rainforests normally too humid to burn.

Campos said 39,000 people in Roraima had been affected either directly or indirectly because of the fires which continued to spread through the region.

“The focal points of fire are increasing and the number of men are out there is insufficient,” Campos said. “The federal government’s structure for fighting fires of this magnitude needs to be more flexible.”

Brazil launched a long-awaited aerial attack on the fires on Sunday, sending two Argentine helicopters armed with huge water buckets to the region of Apiau where fires burning through forest and pastures were threatening homes.

Rain, considered the only effective solution to the crisis, was due to fall in scattered areas in the south of Roraima on Monday but would miss the areas affected by the fires, a forecaster at the National Institute of Meteorology (INMET) said.

More widespread showers would follow on Thursday, but would still not be enough to put out the fires. “It will be very little. It should start raining properly there by April,” said INMET’s Francisco de Assis Diniz.

A thick smoke haze hung over the state capital Boa Vista Monday, reducing visibility to 1.8 miles and forcing aircraft to land with instruments, a spokesman for the city’s international airport said.

A hospital this weekend reported the first fatality from the fires: a three-month-old girl who died after her respiratory illness was aggravated by the smoke.

Some 400 men were combating the fires but the extent of the blazes meant they could not prevent flames from eating ever deeper into the Portugal-sized jungle reservation of the primitive Yanomami Indians.

Reporters and photographers flying over the area last week saw fires advancing into the area and rivers dotted with rocks, indicating water levels were sharply depleted by the drought.

Firefighting experts said the blazes were particularly hard to fight as they were scattered all over Roraima, a state roughly the size of Britain, often creeping through undergrowth in thick jungle difficult to reach by land.

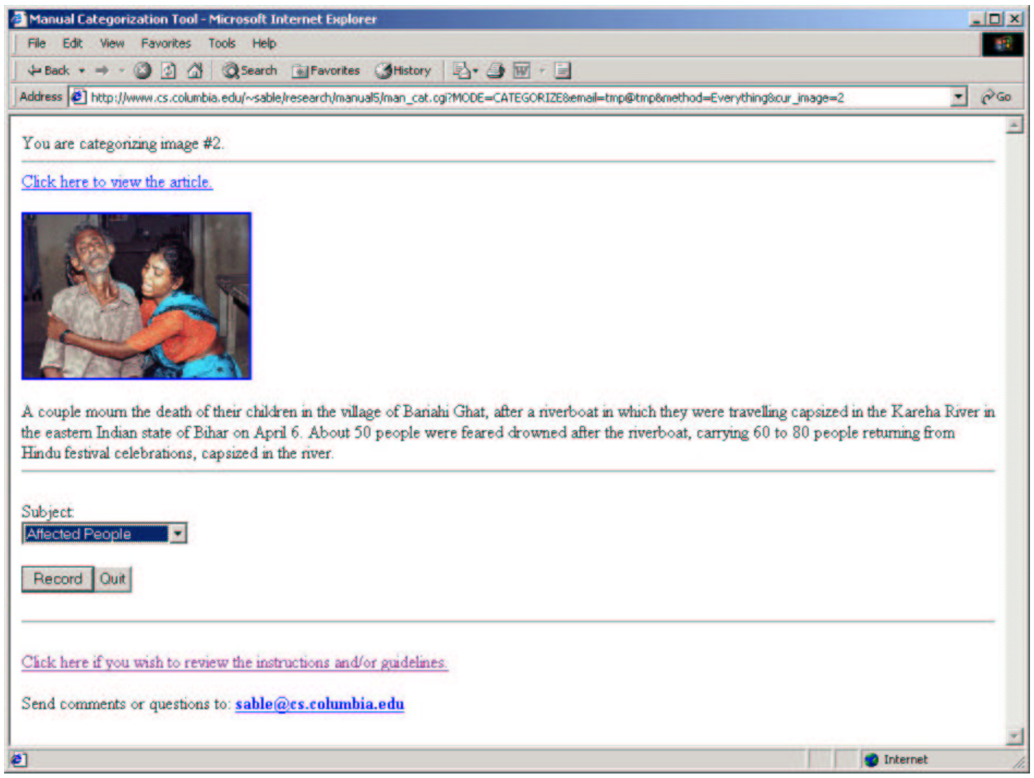
Environmental group Friends of the Earth (FoE) slammed government firefighting efforts on Monday, saying they were ”virtually nonexistent”.

“Without a doubt (government efforts) are insufficient in the sense that they haven’t even properly started,” said Roberto Smeraldi, FoE’s Amazon program coordinator in Brazil. ”There has been no operational response from the authorities.”

APIAU, BRAZIL, 24-MAR-1998: Cordoba, Argentina, firefighters wait for a helicopter to drop water on a forest fire March 24 near Apiau, 120kms (74 miles) south of Boa Vista, capital of Roraima state. Some 120 Argentine firefighters and four helicopters are helping Brazilian counterparts battle the fires which began at the start of March and have been fed by the worst drought since 1926, which has been attributed to the El Nino weather phenomena. [Photo by Marie Hippenmeyer, AFP]

Appendix B

Snapshot of Manual Categorization Tool


Manual Categorization Tool - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address
Go
Record
Quit
Internet

http://www.cs.columbia.edu/~sable/research/manual5/man_cat.cgi?MODE=CATEGORIZE&email=tmp@tmp&method=Everything&cur_image=2

You are categorizing image #2.

Click here to view the article.

A couple mourn the death of their children in the village of Bariahi Ghat, after a riverboat in which they were travelling capsized in the Kareha River in the eastern Indian state of Bihar on April 6. About 50 people were feared drowned after the riverboat, carrying 60 to 80 people returning from Hindu festival celebrations, capsized in the river.

Subject: Affected People

Click here if you wish to review the instructions and/or guidelines.

Send comments or questions to: **sable@cs.columbia.edu**

Appendix C

Instructions for Manually Labeling Documents

For each set of categories that I have defined for my corpus, I have carefully written definitions and guidelines for volunteers who have agreed to manually label images or entire documents. These have been provided along with instructions on how to use the web interface that I have created which allows the volunteers to easily label the images or documents over the web. This interface is described in Section 3.1.2.1, and a snapshot of the interface is shown in Appendix B. In this appendix, I present the instructions for all sets of categories that apply to my corpus.[1]

C.1 Indoor versus Outdoor and Number of People

The research project called CLASS involves automatic categorization of images based on both corresponding text and image features. We need people to manually

[1]I am showing the instructions exactly as they have been provided to volunteers, including spelling errors, grammatical mistakes, and all. Hopefully, there are not too many.

categorize images for two reasons. The first is that we need training sets and testing sets so that we can experiment with our system; Your categorizations help us to define our categories. The second is that we want to evaluate human performance so that we have something to which we can compare our system.

Images are categorized according to one of four features. The four possible features are the image caption, the first sentence of the image caption, the image itself, or the combination of the image with its caption. We ask that you categorize each image using only one of these four features, since we believe that if a single categorizer tries to categorize based on more than one feature, the second categorization might be biased by the first.

We have developed a tool making it easy for you to categorize images for us. The tool is password protected; Send e-mail to sable@cs.columbia.edu to obtain the password if you would like to help us by categorizing images. Then click on the link at the bottom of the page to start the tool. You will be asked to enter your e-mail address, the method you will use to categorize, and the starting image number (from 1 to 1675) that you will categorize. Then, view the appropriate feature for one image at a time, categorize as you see fit, and click "Record" to store your results and move on to the next image. If you make a mistake, you can move back to a previous page (for instance, by using your browsers "Back" button) and categorize an image again. We will only use the last result for each image. You can stop at any time by clicking on "Quit", exiting your web browser, or moving to some other web page. You can use the tool again later if you wish to categorize more images.

Right now, we are dealing with two separate sets of categories. The first is indoor vs. outdoor. With respect to this category set, we are giving you five choices when you see an image feature. They are:

- Indoor

- Likely Indoor
- Ambiguous
- Likely Outdoor
- Outdoor

Here are some guidelines:

1. If you are categorizing based on the image, and the image is partly indoor, partly outdoor, categorize based on what you consider to be the primary focus of the image. For example, if you are shown the image of an interesting room shot from directly outside a window, you can categorize this image as an indoor image.
2. If you are categorizing based on the image, and the image is a close-up of an object and there is no background, you should categorize this image as ambiguous unless you are able to make a logical inference which allows you to choose a category. For instance, a close-up of a wall with graffiti can safely be classified as outdoors, but a close-up of a person's hand would probably be ambiguous.
3. In general, you are allowed to make logical inferences based on the information you are given, and you are allowed to rely on world knowledge, in order to decide your categories. This is why people might beat the machine!

The second category set concerns the number of people in the image. With respect to this set, we are giving you six choices when you see an image feature. The are:

- No people

- One person
- Two people
- Three or more people
- Crowd
- Ambiguous

Here are some guidelines:

1. The "Crowd" category is meant to signify a lage group of people, whereas the "Three or more people" category is for smaller groups. Use your own judgement when deciding between them.
2. The "Ambiguous" category is primarily for those of you who are categorizing based on text. Sometimes, a caption will not give any indication of how many people will be present in an image.
3. When judging based on the image, count a person if most of his body appears in the image, if a small part of his body appears but it takes up a large portion of the image, or if his entire head appears in the image.
4. When judging based only on text, if the description of the image mentions one or two names, then the picture likely contains one or two people respectively. If plurals are used, it is likely a picture of three or more people or a crowd. If the description of the image indicates that it is a picture of an object, there are likely no people. Use your own judgement to determine whether you have enough information to take a guess instead of considering the case to be ambiguous.
5. Count dead bodies the same as live people.

Click here to use the manual categorization tool, and thanks for helping us!

C.2 The Events Categories

How can you help us?

With the rapid expansion of the World Wide Web and the ever-increasing amount of multimedia available, automatic classification of images is a subject of much importance. We are currently dealing with a corpus of news documents containing images with corresponding captions and articles, and we are exploring methods to categorize the documents into one of a few general news categories and then to ask the question of how the image relates to the story.

In order to automatically categorize news documents, we need to show our system sample documents for each category. In order to evaluate our system, we need to test it with labeled documents. For both of these reasons, we need people to manually categorize some of our documents.

We have developed a tool allowing volunteers to categorize news documents over the web. If you would like to help us, please send email to sable@cs.columbia.edu. We will then provide you with a few final details and assign you a starting document number so that people won't all end up categorizing the same documents! You will then be ready to click at a link at the bottom of this page to start using the tool. You will be asked to enter a start index (document number of the first document to be categorized), the method used to categorize documents (either article only or article plus image and caption), and your e-mail address.

After you verify that the information you have entered is correct, you will then be presented with one news document at a time. Since the articles are large, they are not displayed in the html forms themselves; rather, a link to the article for each document is displayed. The articles contain most of the documents' content, so please click on each article link to view each article. You don't have to read the entire article, but please read the start of each and skim the rest; whatever you feel is necessary to get a good idea of what the article is about. Then click on the "Back" button of your browser to view the document's image and caption and choose your categorization. Click "Record" to store your categorization for each document and move on to the next document. If you make a mistake, you can move back to a previous page (for instance, by using your browsers "Back" button) and categorize a document again. We will only use your last categorization for each individual document. Each document will also contain a link to this page in case you wish to review the instructions or guidelines. After reviewing, move back to the page with the document to choose and record your categorization. You can stop at any time by clicking on "Quit", exiting your web browser, or moving to some other web page. You can use the tool again later if you wish to categorize more images.

What are the categories?

We are asking you to categorize each news document into one of five categories: Struggle, Politics, Crime, Disaster, or Other. We are defining these categories as follows:

- Struggle - News documents to be placed in this category include those concerning international wars, civil wars, civil unrest such as protests and demonstrations, acts of political terrorism that are part of an ongoing conflict in a particular area, plights of refugees, etc. These documents focus on events or conditions which are intrinsic to struggle.

- Politics - News documents to be placed in this category include those about campaigns, elections, political meetings or debates, political press conferences, personal affairs of politicians, obituaries of politicians, political crimes, etc. These documents focus on political activity or politicians.

- Crime - News documents to be placed in this category include those concerning violent crimes, non-violent crimes, individual acts of terrorism that are not part of an ongoing struggle, acts of mindless violence, etc. These documents may focus on the act itself or possibly investigations, trials, or other consequences of the crime.

- Disaster - News documents to be placed in this category include those about natural disasters and accidents. These documents generally focus on devastation.

- Other - Any document that does not fit well into one of the other four categories should be placed here.

Below are some guidelines to help you get a feel for these categories. It is very important that you read them carefully before beginning to categorize documents:

- Some articles may seem to fit into multiple categories. In cases where the categorization of a document seems ambiguous, we ask you to determine the main focus of the document, and choose the category that gives the best fit.

- If deciding between one of the first four categories and the Other category, again ask yourself what is the main focus of the document, and try to determine if it fits in well with the other documents in the major category and with our definitions above. If not, place it in Other.

- We understand that these categories are not entirely distinct in their natures. For instance, most struggles are political, and a Struggle category could almost be considered a subcategory of a Politics category. This is fine; we have chosen the above categories for reasons that will be discussed further below.

- Documents that focus on political meetings, political debates, or political statements about struggles should be placed in the Politics category. Documents that focus on a struggle but mention related political aspects or issues should be placed in the Struggle category. Again, you need to ask yourself what is the focus of the document.

- It may seem a bit unusual that the definitions above do not place all documents concerning terrorism into the same category. We have found through examination of many documents that stories about political terrorism which is part of an ongoing struggle, including topics such as bombings by the PLO or HAMAS, have more in common with other documents in the Struggle category, while stories concerning individual acts of terrorism, including topics such as the Oklahoma City bombing or The Unabomer, have more in common with other documents in the Crime category (e.g. such events are often followed by investigations and trials). In addition, the images contained within these documents fit better into the corresponding categories, and, once again, we have chosen these categories for reasons discussed below.

- The magnitude of an event on which a document focuses should not generally affect the document's categorization. For example, a document about the effects of a storm should be placed in the Disaster category even if there were no deaths or injuries.

- We fully expect that you will often need to rely on your intuition when categorizing a document. If you find that the definitions and guidelines above seem to indicate one category but you strongly feel that a document should be placed in another, just use your own judgement, and make what you consider to be the best choice. This is why we have multiple human volunteers, and in the end, one of the things we wish to measure is level of agreement between different people. In the end, only documents with agreement between multiple categorizers will be used to define categories.

Why have we chosen these categories?

We hope to gain two things from the automatic categorization of news documents. First, we want the categorizations to be useful in and of themselves. Therefore, the categories must be understandable and intuitive, and of a nature such that there might be some interest in narrowing searches or browsings of documents to one or more of these categories. Second, we want the categorization of a document to help us in answering the question of how a document's image is related to the story. For example, if we know that a document is categorized in the Disaster category, we might expect that the image will be one of wreckage, rescue workers, victims, or mourners. Some of our guidelines above were based on this desire; for instance, we could have asked for documents concerning political meetings about struggles to be placed in the Struggle category, but instead, we ask for them to be placed in the Politics category. This is because we have made the observation that the majority of such documents contain images of politicians! Two other things to consider are that we need categories such that our system is likely to achieve high accuracy. We believe that with a few categories containing documents with similar content, this will be the case. Finally, we need each category to be sizeable enough such that we will have enough training documents to represent it well, without being so large that it will be uninteresting to specify it. Perusal of a few hundred images leads us to believe that the four major categories above will contain between 90% and 95% of the documents in our corpus, and that each individual category will fit all of the requirements mentioned here.

Are you ready to categorize?

Click here to use the manual categorization tool, and thank you for helping us!

C.3 The Disaster Image Categories

Why do we need volunteers?

With the rapid expansion of the World Wide Web and the ever-increasing amount of multimedia available, automatic classification of images is a subject of much importance. We are currently dealing with a corpus of news documents containing images with corresponding captions and articles. We have previously explored methods to categorize each document into one of a few general news categories. We are now exploring methods to categorize the images these documents contain based on their depicted content.

In order to automatically categorize images, we need to show our system sample images for each category. In order to evaluate our system, we need to test it with labeled images. For both of these reasons, we need people to manually categorize some of our images.

What are the categories?

We have previously trained our system to categorize news documents into five categories which we have called Struggle, Politics, Crime, Disaster, and Other. We defined these categories to be mutually exclusive. We are now focusing on the images contained in those documents which were labeled as Disaster documents (based upon agreement between multiple humans). The documents in that category concerned natural disasters and accidents, and commonly focused on devastation.

We are asking you to examine the images contained in these documents and to place each image into one of the following four categories:

- Affected People - Images showing one or more people who were in some way affected by the disaster. These include direct victims (dead or alive) as well as friends or relatives of victims (e.g. mourners).

- Workers Responding - Images showing people responding to a disaster, perhaps to aid victims. Such people include rescue workers, fire fighters, police, etc. The people do not necessarily need to be ”professional” workers.
- Wreckage - Images showing some or all of the damage caused by the disaster.
- Other - Any image which does not fit well into the preceeding three categories.

Below are some important guidelines to keep in mind when categorizing images. Please read them carefully before you begin.

- Although you are categorizing images, it is fine for you to use the captions to aid your decisions, and this may often be necessary. For example, if an image contains a person, you may need to read the caption (if it isn't obvious from the image itself) to determine who the person is, and whether it is someone affected by the disaster or a worker responding to it.
- Some articles may seem to fit into multiple categories. In cases where the categorization of an image seems ambiguous, we ask you to determine the main focus of the image, and choose the category that gives the best fit. For example, many images contain both affected people or workers responding and also wreckage. If the focus of the image is on the person or people, and they happen to be pictured in the setting of wreckage, place the image in the one of the first two categories. If the focus of the image is the wreckage, and there happens to be affected people or workers responding in the background, place the image in the third category. At times, you may want to use the caption to help determine the focus of an image.

- We fully expect that you will often need to rely on your intuition when categorizing an image. If you find that the definitions and guidelines above seem to indicate one category but you strongly feel that an image should be placed in another, just use your own judgment, and make what you consider to be the best choice. This is why we have multiple human volunteers, and in the end, one of the things we wish to measure is level of agreement between different people. In the end, only images with agreement between multiple categorizers will be used to define the categories.

How do you categorize?

We have developed a tool allowing volunteers to categorize images over the web. If you would like to help us, please send email to sable@cs.columbia.edu. We will then provide you with a few final details and assign you a starting image number so that people won't all end up categorizing the same images! You will then be ready to click at a link at the bottom of this page to start using the tool. You will be asked to enter a start index (image number of the first image to be categorized), the method used to categorize images (either viewing everything or only the caption - most volunteers will select the first of these two options), and your e-mail address.

After you verify that the information you have entered is correct, you will then be presented with one image at a time along with its caption. If you wish, you can click on an image to view a (usually) larger version, then click on the "Back" button of your browser to return to the previous page. (Since the articles are large and generally will not be necessary for you to choose your categorizations, they are not displayed in the html forms themselves; however, a link to the article associated with each image is provided. If you want to view an article, click on this link. When you are finished with the article, click on the "Back" button of your browser return to the image.) Once you have made your decision, select the chosen category in the provided drop-down, and then click "Record" to store your categorization of the image and move on to the next image. If you make a mistake, you can move back to a previous page (for instance, by using your browser's "Back" button) and categorize an image again. We will only use your last categorization for each individual image. With each image, you will also be provided a link to this page in case you wish to review the instructions or guidelines. After reviewing, move back to the page with the image to choose and record your categorization. You can stop at any time by clicking on "Quit", exiting your web browser, or moving to some other web page. You can use the tool again later if you wish to categorize more images.

Are you ready to categorize?

Click here to use the manual categorization tool, and thank you for helping us!

C.4 The Politics Image Categories

Why do we need volunteers?

With the rapid expansion of the World Wide Web and the ever-increasing amount of multimedia available, automatic classification of images is a subject of much importance. We are currently dealing with a corpus of news documents containing images with corresponding captions and articles. We have previously explored methods to categorize each document into one of a few general news categories. We are now exploring methods to categorize the images these documents contain based on their depicted content.

In order to automatically categorize images, we need to show our system sample images for each category. In order to evaluate our system, we need to test it with labeled images. For both of these reasons, we need people to manually categorize some of our images.

What are the categories?

We have previously trained our system to categorize news documents into five categories which we have called Struggle, Politics, Crime, Disaster, and Other. We defined these categories to be mutually exclusive. We are now focusing on the images contained in those documents which were labeled as Politics documents (based upon agreement between multiple humans). The documents in this category concern campaigns, elections, political meetings or debates, political press conferences, personal affairs of politicians, obituaries of politicians, political crimes, etc. These documents usually focus on political activity or politicians.

We are asking you to examine the images contained in these documents and to place each image into one of the following six categories:

- Meeting - Images to be placed in this category include those that depict meetings taking place or getting started, as well as images which show politicians arriving to or leaving from a meeting or talking together during a break. The meetings will often be official, for example summits or congressional gatherings, but they can be casual or informal. Images which show politicians making an announcement about the outcome of a meeting belong to the next category.

- Announcement - Images to be placed in this category include those depicting dispersal of any type of political information, perhaps in the form of a press conference, speech, ceremony, or interview. These announcements may also be informal, for example if a politician starts talking to the press outside his or her house.

- Politician Photographed - Images which show a politician in daily life, either on the job or performing personal activities. If, however, the politician has started talking to the press and answering questions, the image might be placed in the previous category instead.
- Civilians - These images show individuals or groups of civilians in some country related to the political story. These include pictures of protests and demonstrations. This does not include workers related to the story; for example, images showing soldiers or police would be included in the next category.
- Military - Images to be placed in this category include pictures of military personnel, police, and peace-keeping forces. They can be in action or preparing for action. This category can also include pictures of vehicles, such as military tanks or aircrafts. It is OK if a politician is also in the photograph.
- Other - Any image which does not fit well into the preceeding five categories.

Below are some important guidelines to keep in mind when categorizing images. Please read them carefully before you begin.

Although you are categorizing images, it is fine for you to use the captions to aid your decisions, and this may often be necessary. For example, if an image contains a person, you may need to read the caption (if it isn't obvious from the image itself) to determine who the person is and what the person is doing.

- Some articles may seem to fit into multiple categories. In cases where the categorization of an image seems ambiguous, we ask you to determine the main focus of the image, and choose the category that gives the best fit.
- We fully expect that you will often need to rely on your intuition when categorizing an image. If you find that the definitions and guidelines above seem to indicate one category but you strongly feel that an image should be placed in

another, just use your own judgment, and make what you consider to be the best choice. This is why we have multiple human volunteers, and in the end, one of the things we wish to measure is level of agreement between different people. In the end, only images with agreement between multiple categorizers will be used to define the categories.

How do you categorize?

We have developed a tool allowing volunteers to categorize images over the Web. If you would like to help us, please send email to sable@cs.columbia.edu. We will then provide you with a few final details and assign you a starting image number so that people won't all end up categorizing the same images! You will then be ready to click at a link at the bottom of this page to start using the tool. You will be asked to enter a start index (image number of the first image to be categorized), the method used to categorize images (either viewing everything or only the caption - most volunteers will select the first of these two options), and your e-mail address.

After you verify that the information you have entered is correct, you will then be presented with one image at a time along with its caption. If you wish, you can click on an image to view a (usually) larger version, then click on the "Back" button of your browser to return to the previous page. (Since the articles are large and generally will not be necessary for you to choose your categorizations, they are not displayed in the html forms themselves; however, a link to the article associated with each image is provided. If you want to view an article, click on this link. When you are finished with the article, click on the "Back" button of your browser to return to the image.) Once you have made your decision, select the chosen category in the provided drop-down, and then click "Record" to store your categorization of the image and move on to the next image. If you make a mistake, you can move back to a previous page (for instance, by using your browser's "Back" button) and categorize an image again. We will only use your last categorization for each individual image. With each image, you will also be provided a link to this page in case you wish to review the instructions or guidelines. After reviewing, move back to the page with the image to choose and record your categorization. You can stop at any time by clicking on "Quit", exiting your web browser, or moving to some other web page. You can use the tool again later if you wish to categorize more images.

Are you ready to categorize?

Click here to use the manual categorization tool, and thank you for helping us!

Appendix D

Labels for the Indoor versus Outdoor Data Set

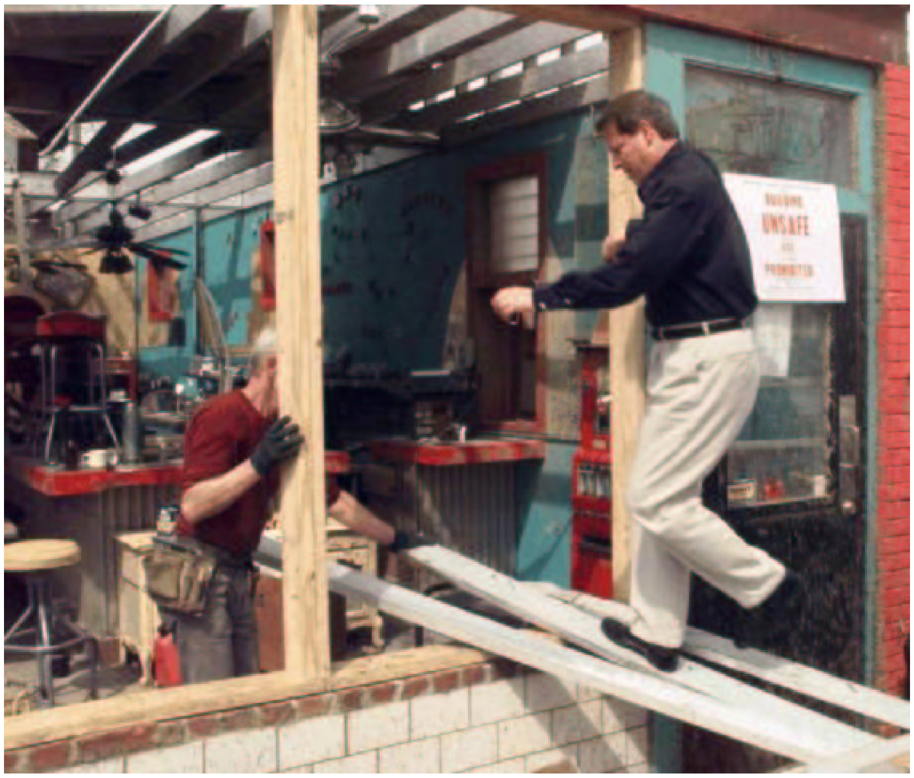
As described in Section 3.1.2.2, for the *Indoor* versus *Outdoor* data set, a total of three volunteers have labeled 1,675 images, and I have labeled the same 1,675 images. All labelers have been shown the image along with its caption in order to make their decisions. Five choices exist for each image: *Indoor*, *Likely Indoor*, *Ambiguous*, *Likely Outdoor*, and *Outdoor*. The instructions that have been provided for these categories are presented in Appendix C.1 and can also be found at http://www.cs.columbia.edu/~sable/research/readme.html. 1,339 (79.9%) of the 1,675 images have been assigned a definite decision in the same direction by both me and the volunteer, and these 1,339 images comprise the primary data set used for the experiments with the *Indoor* and *Outdoor* categories discussed in this thesis. 401 (29.9%) of these images are classified as *Indoor* and 938 (70.1%) are classified as *Outdoor*. Although researchers who want to compare results directly with mine would have to use the same data set, I am providing all of the labels assigned by volunteers with the public release of my corpus, so researchers who want to use less strict definitions of categories or experiment with ambiguous images can do so.

Other Evaluation	My Evaluation				
	Indoor	Likely Indoor	Ambiguous	Likely Outdoor	Outdoor
Indoor	401	15	7	4	4
Likely Indoor	47	12	3	5	3
Ambiguous	21	9	14	9	7
Likely Outdoor	13	10	2	12	10
Outdoor	35	14	14	66	938

Table D.1: This table shows how 1,675 images have been labeled by me and by volunteers for the *Indoor* versus *Outdoor* data set.

Table D.1 shows how labels for the 1,675 images have been assigned by the volunteers and by me. The columns represent my own labels and the rows represent labels chosen by volunteers. The numbers in the upper left cell of the table and the lower right cell of the table represent the 1,339 images that have been assigned definite decisions in the same direction by both me and the volunteer. The other images had varying levels of disagreement, or some had agreement but not about absolute category decisions.

You can see that 14 of the images have been labeled as *Ambiguous* by both me and a volunteer, and another 23 have been labeled as *Ambiguous* by one person with the second person showing a non-absolute preference in one direction or the other. All 37 of these images can be seen at http://www.cs.columbia.edu/~sable/research/io_ambiguous.html. There are several reasons for ambiguity for this data set. Some images show scenes that are partially inside and partially outside; some images have lighting that makes it impossible to tell (for some but not all of these cases, the text might make it obvious); some images show action taking place under the roof of a structure without walls, or in between the walls of a building with a collapsed roof; some images are of icons, maps, or text that do not seem to fit in this domain all together. One such image that has been labeled as *Ambiguous* by both me and



NASHVILLE, TENNESSEE, 17-APR-98: Vice President Al Gore climbs through a broken window April 17 to view the tornado damage done to Joe's Diner in Nashville. Two tornadoes hit Nashville on April 16, injuring over 100 people and damaging at least 500 buildings. [Pool Photo by Mark Humphrey, Reuters]

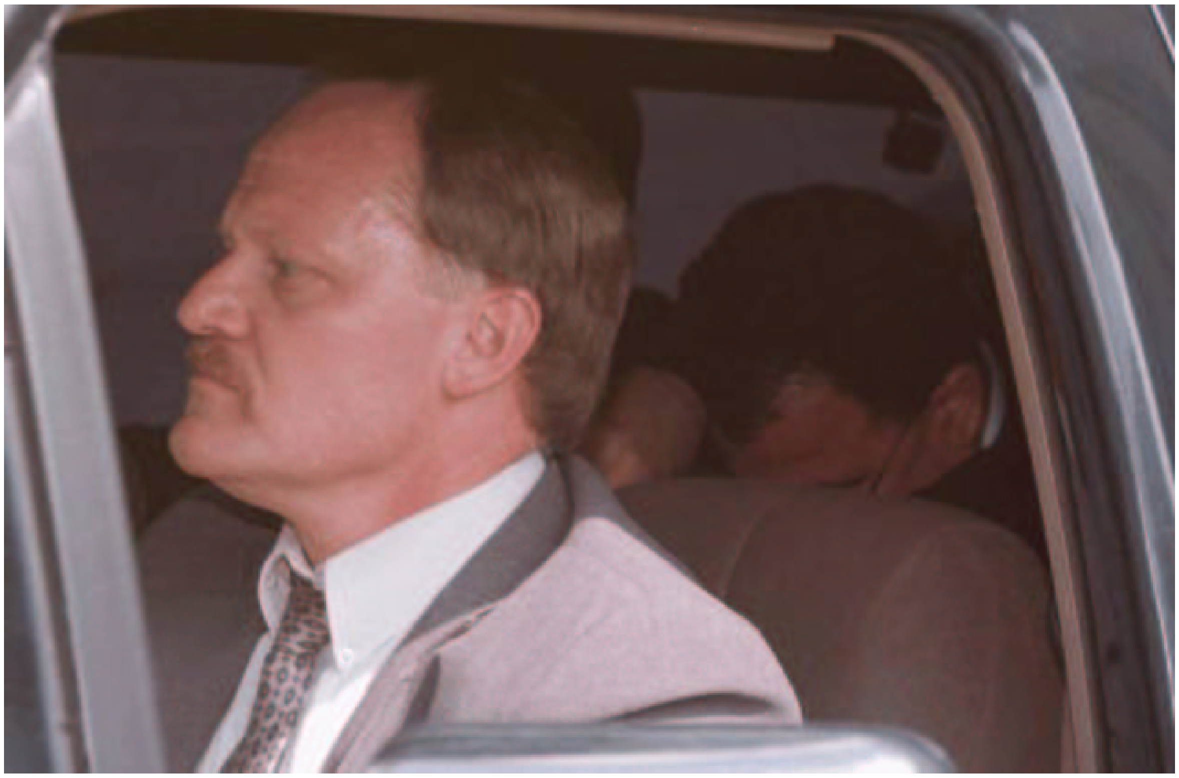
Figure D.1: This image is partially *Indoor* and partially *Outdoor*, not to mention the fact that the ceiling of this shop appears to be missing, making this image hard to label (as agreed by me a volunteer).

a volunteer is shown in Figure D.1.

You can also see that for several images, there is serious disagreement between me and a volunteer. There are 137 images that differ by two or more steps on the scale from *Indoor* to *Outdoor*, and 39 of these are cases for which one of us said *Indoor* with certainty and the other person said *Outdoor* with certainty. These 137 images can be seen at http://www.cs.columbia.edu/~sable/research/io_disagreement.html. An analysis of these cases has revealed that some of these differences are due to obvious mistakes by either me or the volunteer.[1] Other differences, however, are due to honest disagreements concerning the definitions of the categories. Images that have led to such philosophical disagreements include drawings (as opposed to photographs) of scenes that are clearly *Indoor* or *Outdoor*; close-ups of individuals inside vehicles such as cars, planes, or trains; and images taking place in unusual settings, e.g. underwater. One image that I have labeled as clearly *Indoor* but a volunteer has labeled as clearly *Outdoor* is shown in Figure D.2. Interestingly, you can see from Table D.1 that extreme disagreement in this direction has happened far more often (35 times) than the other way around (4 times).

This appendix shows that labeling images can be confusing. I would have expected that these particular categories (*Indoor* and *Outdoor*) would be relatively intuitive and simple. In creating my corpus, however, I have discovered that no matter how much time is spent defining and providing guidelines for categories,

[1]The first version of the manual categorization tool, which has been used for these categories, provides a radio button for each category. The buttons are mutually exclusive, so only one category can be selected at a time. When a user would move from one document to the next, the initially selected category would be the category selected for the previous document. If a user would click on a new label and then submit the decision quickly, it is possible that the original click would be missed by the interface without the user noticing, in which case the label from the previous document would be mistakenly used. I estimate that this has occurred for somewhere between 1% and 2% of the labels for this data set. For future data sets, I have replaced the radio buttons with drop-down boxes, and the initially selected category is always the generic *No Selection*. Since implementing this change, I have not noticed any obvious mistakes by any human labeler.



WASHINGTON, USA, 18-JUN-97: Saudi dissident Hani al-Sayegh sits in the back of the car (R) after leaving the U.S. Federal District Courthouse June 18, where he was charged with conspiracy to commit murder and "international terrorism" in connection with the deadly truck bombing against U.S. troops in Saudi Arabia. Sayegh was deported from Canada and has agreed to cooperate with the U.S. investigation of last June's bombing that killed 19 U.S. airmen. An unidentified driver is at left. [Photo by Mark Wilson, Reuters]

Figure D.2: This image clearly takes place in an enclosed space (a car), and I consider it to be an *Indoor* image, but the car (and probably the camera) are outside, so the volunteer who has labeled it considers it to be an *Outdoor* image.

there are always cases that are confusing, and cases for which there is honest disagreement. Appendix E shows that the same thing is true for the Events data set. You can see in Section 3.1 that for every data set in my corpus, the portion of images that are assigned definite labels in the same direction is under 85% for all four data sets, and under 80% for three of the four data sets. This is one of the reasons that it is difficult to create a new text categorization corpus.

Appendix E

Labels for the Events Data Set

As described in Section 3.1.2.3, for the data set involving the Events categories, 28 volunteers have labeled 1,750 documents, and I have labeled the same 1,750 documents. This time, evaluators have been asked to categorize entire documents, each consisting of an image, caption, and article. The choices are the categories themselves: *Struggle*, *Politics*, *Disaster*, *Crime*, and *Other*. The instructions that have been provided for these categories are presented in Appendix C.2 and can also be found at http://www.cs.columbia.edu/˜sable/research/instr.html. A total of 1,328 (75.9%) of the 1,750 documents have been assigned identical labels by both me and the volunteer, and these 1,328 documents comprise the data set used for the Events categories. 417 (31.4%) of these documents are classified as *Struggle*, 387 (29.1%) are classified as *Politics*, 296 (22.3%) are classified as *Disaster*, 150 (11.3%) are classified as *Crime*, and 78 (5.9%) are classified as *Other*.

Table E.1 shows how labels for the 1,750 documents have been assigned by the volunteers and by me. The columns represent my own labels and the rows represent labels chosen by volunteers. The numbers along the diagonal running from the top left to the bottom right of the table represent the 1,328 documents that have been assigned identical labels by both me and the volunteer; the other

Other Evaluation	My Evaluation				
	Struggle	Politics	Disaster	Crime	Other
Struggle	417	77	2	24	35
Politics	84	387	4	18	65
Disaster	8	0	296	2	7
Crime	33	7	2	150	9
Other	7	12	18	8	78

Table E.1: This table shows how 1,750 documents have been labeled by me and by volunteers for the Events data set.

documents have been assigned two distinct labels. Note that most of the table is approximately symmetric, but there is a large exception. There are 100 documents that I have classified as *Other* but a volunteer has classified as either *Struggle* or *Politics*, whereas the opposite disagreements have occurred a total of only 19 times. This indicates that I have probably been more willing to label a document as *Other* than most of the volunteers.

category	Agreement for category
Struggle	60.7%
Politics	59.2%
Disaster	87.3%
Crime	59.3%
Other	32.6%

Table E.2: Humans tend to agree the most for the *Disaster* category and the least for the *Other* category.

One interesting statistic which can be calculated based on the information provided in Table E.1 is the probability that a document is assigned to a category given that at least one label for that document indicates the category. In other words, for each category, given that a document is assigned at least one label indicating the category, it is possible to calculate the probability that both labels indicate the category. This can be achieved by dividing the number of documents assigned two labels indicating the category by the number of documents assigned at least one label indicating the category. These probabilities can serve as a measure of the level of human agreement for each category. Table E.2 shows the level of human agreement computed using this metric for each of the five Events categories. The *Disaster* category has the highest level of agreement by far and the *Other* category has the lowest level of agreement by far; the other three categories have very similar levels of agreement.

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
My systems						
BINS	88.0	87.5	88.3	97.2	83.1	60.0
Naive Bayes	86.0	85.4	87.2	96.7	81.4	27.6
Rocchio/TF*IDF	87.1	85.0	88.4	98.8	79.2	60.0
Density Estimation	84.9	83.7	86.0	97.3	80.0	34.3
K-Nearest Neighbor	84.0	81.1	82.1	93.9	81.3	65.0
Rainbow systems						
Naive Bayes	87.6	86.2	86.3	96.7	89.1	61.5
Rocchio/TF*IDF	87.4	81.1	85.3	97.7	88.4	68.3
K-Nearest Neighbor	81.9	80.0	79.7	95.6	75.6	63.2
Probabilistic Indexing	86.5	83.6	84.8	97.2	89.4	65.0
Support Vector Machines	88.7	88.1	89.2	96.2	87.0	57.9
Maximum Entropy	88.3	88.1	87.9	95.7	87.9	55.6

Table E.3: Text based systems perform the best for the *Disaster* category and the worst for the *Other* category.

Table E.3, which is identical to Table 5.5, shows the results of many text based systems that have been tested for the Events data set. For the purposes of this appendix, the first column of results (displaying the overall accuracy of each system) is not important. The remaining columns show the performance of each system for individual categories using the F_1 measure, first explained in Section 2.7.

Note that every system, without exception, performs the best by far for the *Disaster* category and the worst by far for the *Other* category. The ranking of performance for the other three categories (*Struggle*, *Politics*, and *Crime*) varies, with systems generally achieving similar performance for all three.

I have shown that, for the Events categories, text based systems do the best (or worst) for the same category that humans have the highest (or lowest) level of agreement. While this may seem intuitive, there are at least two reasons why it does not have to turn out this way. First, only documents with agreement are included in the final training and test sets, so it is not clear that documents ultimately placed in a category that has a low level of human agreement must be relatively ambiguous. Second, the methods that statistical systems use to perform automatic categorization are presumably very different from the methods humans are using when they label documents. Humans are understanding the "gist" of the document, and they are relying on intuition and a lot of real world knowledge to make their decisions, while the systems are all using purely statistical techniques involving bag of words approaches.

It might be interesting to carry out a study to see if a similar result holds for other data sets (both in my own corpus and otherwise). Furthermore, for the Events data set (and perhaps others), one might try to estimate the level of correlation between system performance for a category and the level of human agreement for the category; one might also try to determine if there is a correlation between the types of errors that systems tend to make and the types of human disagreements that tend to occur. I leave such a study for future work.

Appendix F

The Number of People Data Set

This set of categories has been defined largely because the researchers at Columbia who were working with image features wanted them. People detectors and face detectors have received significant attention in that literature (e.g. see (Srihari, 1995)). In my own research, this set of categories has not been used as much as the others that I have defined because it is very difficult to determine this automatically with text categorization (at least not with standard approaches - it is possible that a deeper NLP based approach can be successful). However, there is one interesting experiment in which I utilize these categories, which I mention at the end of this appendix.

The same three volunteers that have labeled images for the *Indoor* versus *Outdoor* data set defined in Section 3.1.2.2 have also labeled the same 1,675 images according to the number of people present in the image, and I have done this as well. The choices given are the categories themselves, which are: *No People*, *One Person*, *Two People*, *Three or More People*, *Crowd*, and *Ambiguous*. The distinction between the *Crowd* category and the *Three or More People* category is left vague, and this is largely left to the opinion of the labeler. The instructions that have been provided for these categories are presented in Appendix C.1 and can also be found at http://www.cs.columbia.edu/~sable/research/readme.html. 1,346 (80.4%) of the 1,675 images have been assigned identical, non-ambiguous judgments from both me and the volunteer, and these comprise the data set used for these categories. 88 (6.5%) of these images are classified as having no people, 304 (22.6%) have one person, 213 (15.8%) have two people, 609 (45.2%) have three or more people, and 132 (9.8%) depict a crowd.

Although this data set has been defined for researchers working with image features, I have used it for one experiment that I and a co-author discuss in Section 9 of (Sable and Hatzivassiloglou, 2000) involving the integration of separate sets of categories. It turns out that there is a high correlation between the number of people in an image and the *Indoor* versus *Outdoor* setting. For example, images with no people or with crowds are almost always *Outdoor*, whereas images with exactly one or two people are about equally likely to be *Indoor* or *Outdoor*. (Remember, *Outdoor* images account for about 70% of the *Indoor* versus *Outdoor* data set.) The experiment involves updating predictions for the *Indoor* versus *Outdoor* data set using information about the number of people in each image. It turns out that we only achieve a minor improvement in overall accuracy. However, using an alternate evaluation metric that gives partial credit for right or wrong answers based on confidence, we see a more significant improvement. In other words, while few categorizations change from right to wrong or vice versa, correct decisions are, on average, given higher confidence, while the reverse happens for incorrect decisions. I will not discuss this experiment further here, but the paper cited earlier in this paragraph describes it in detail.

Appendix G

Testing Features with Cross Validation

The first text categorization system that I implemented was an augmented Rocchio/TF*IDF system (this methodology is described in Section 2.6.1.1). I have experimented with several features, some of which are novel (e.g. the inclusion of parts of speech as part of word tokens, or the use of various spans of text). By testing these features, I have stayed within a bag of word paradigm with the goal of examining the effects that various features have on performance. In order to determine which features are the most significant for achieving good performance, I have relied on cross validation experiments within the training set using all possible combinations of feature values (cross validation is explained in Section 2.5). The results of such cross validation experiments when applied to an older version of the *Indoor* versus *Outdoor* data set are reported in detail in (Sable and Hatzivassiloglou, 2000); in this appendix, I summarize the results of the cross validation experiments using the training set of the current version of the *Indoor* versus *Outdoor* data set (described in Section 3.1.2.2) as well as the training set of the Events data set (described in Section 3.1.2.3).

G.1 The Features

My Rocchio/TF*IDF system allows the user to vary the values of seven optional parameters. I have set up an automatic script that applies three-fold cross validation within the training set, and this is repeated for every possible combination of values of parameters. Based on the results of these cross validation experiments, the best values of parameters can be selected, and training can be repeated with these settings using the entire training set. Five of the optional parameters discussed in this appendix relate to the definition of a word, one deals with normalization (optional for the Rocchio technique), and the last is density estimation (discussed in Chapter 4). More specifically, the parameters varied in the cross validation experiments for the *Indoor* versus *Outdoor* data set and the Events data set are as follows:

- **Text span considered**. What text should be considered when predicting a category for each image? I have experimented with using the entire article and image caption concatenated, the article without the caption, just the caption, and only the first sentence of the caption. While the articles are longer and provide more information about the related story than the caption, they are less related to the specific image, and therefore may contain too much noise to be helpful for the certain types of categorization. Similarly, in our corpus, the first sentence of the caption tends to be more descriptive of the image than the rest, which often provides background information. What type of information is appropriate may depend on the specific categories with which we are dealing.

- **Restriction to specific grammatical categories**. Should all the words in the selected text span be included in the TF*IDF computations? Open-class words (i.e., adjectives, nouns, verbs, and adverbs) tend to carry most of the content information, while words such as numbers and pronouns do not usually affect an image's classification. I used Church's statistical part-of-speech tagger (POS) (Church, 1988) to automatically assign a grammatical category tag to each word. For the cross validation experiments applied to the *Indoor* versus *Outdoor* training set, I experimented with using all words, only open-class words, and open-class words excluding proper nouns. For the cross validation experiments applied to the Events training set, I also tested using just nouns and nouns excluding proper nouns, since it has been suggested that nouns might be particularly important for these categories.

- **Disambiguation of words**. A word's sense is frequently ambiguous, and sometimes knowing its grammatical part-of-speech can help disambiguate it. For example, the word "can" is most often an auxiliary verb, but sometimes a noun with a different meaning. I have experimented with including the POS tag as part of the word (thus distinguishing between the two senses of "can" - *can/modal* versus *can/noun* - in this example), versus ignoring this information.

- **Case sensitivity**. Should capitalization matter for treating words as different? Capitalization may indicate a proper noun, but may also be the result of sentence-initial placement. I experimented with collapsing words that differ only in capitalization to the same token versus treating words as different if they differ in case.

- **Thresholds for TF*IDF values**. I have experimented with optionally ignoring words whose TF*IDF values within a document fall below a given constant, for several alternative values of that constant. This eliminates relatively insignificant words, which have minimal (or negative) impact on the classification, and is similar to using a stop-word list, as discussed in Section 2.4.1. For my cross validation experiments, I test three thresholds, which I call low, medium, and high, and I also test not using any threshold.

- **Normalization of category vectors**. The number of documents in various categories can differ substantially, as is the case with the *Indoor* versus *Outdoor* data set and the Events data set. It is natural to expect that the *a priori* most frequent category would have higher TF values in category vectors, simply because it contains more documents. This is a concern for these experiments, since, for example, the *Outdoor* category contains more than two thirds of the images in my collection; however, it is not always bad to give an "advantage" to the larger category. I have therefore repeated the cross validation experiments both with and without normalization of the category vectors.

- **Density estimation**. The standard approach for assigning a document to a category is to select the category for which the similarity is the largest. This, however, implicitly assumes that the similarity scores are on the same scale for all categories, and makes it hard to tell when a difference between the similarity scores for the two categories is large enough for the system to be confident in its decision. I have experimented with an alternative method of choosing a category based on similarity scores. Density estimation is an established statistical technique (Silverman, 1986) for estimating a probability density for the distribution assumed to generate a set of empirically obtained data points. I describe my approach to using density estimation for text categorization in detail in Chapter 4 (and also in (Sable, McKeown, and Hatzivassiloglou, 2002)), and I do not go in to any more detail here. For my cross validation experiments, I test three possible window sizes, which I refer to as small, medium, and large, and I also test not using density estimation at all (i.e. choosing the category with the highest similarity score instead).

G.2 Cross Validation Results for the Indoor versus Outdoor Data Set

There are 1,536 possible combinations of the 7 optional parameters as described above for the *Indoor* versus *Outdoor* data set. Every one of these combinations has been tested using three-fold cross validation on the training set of this data set. The settings of certain parameters turn out to be significantly more important than the settings of others.

Table G.1 shows the settings used for the optional parameters for the best 11 cross validation results for the training set of the *Indoor* versus *Outdoor* data set. These cross validation experiments all achieve an overall accuracy of approximately 83% or greater, with the single best cross validation experiment obtaining an overall accuracy of approximately 83.8%. Table G.2 shows the average overall accuracy of all cross validation experiments for each possible value of all these parameters. Usually, the settings that lead to the highest average over all experiments in which they are used also tend to show up in the majority of the best experiments, but this is not always the case (exceptions are noted below). Sometimes, settings of various parameters do well when used together but not alone, and certain parameters may be more important with specific settings of other parameters. For example, normalization and density estimation can both account for skewed category sizes, and it is possible that one might be more important if the other is not used. In any case, when choosing parameter settings for training based on the entire training set (to train the system for the test set, or for future documents), it is probably more important to use settings from the best experiments, as this is the performance we want to emulate. The effects of each of the various parameters varied in the cross validation experiments using the *Indoor* versus *Outdoor* data set can be summarized as follows:

Rank	Text Span Used	Part-of-Speech Used	Include Tag	Case Sensitive	Thresholds	Normalize	Density Estimation
1	first sentences	open-class words except proper nouns	yes	no	medium	no	small window
2	first sentences	open-class words except proper nouns	yes	yes	medium	no	small window
3	first sentences	open-class words except proper nouns	yes	no	medium	yes	small window
3	first sentences	open-class words except proper nouns	yes	yes	medium	no	large window
5	first sentences	open-class words except proper nouns	yes	yes	medium	yes	small window
5	first sentences	open-class words except proper nouns	yes	yes	medium	no	medium window
7	first sentences	all words	yes	yes	medium	no	medium window
7	first sentences	open-class words except proper nouns	yes	no	medium	no	medium window
9	first sentences	open-class words except proper nouns	yes	yes	low	yes	medium window
9	first sentences	open-class words except proper nouns	no	yes	low	yes	medium window
9	first sentences	open-class words except proper nouns	no	yes	medium	no	small window

Table G.1: Some parameters have specific settings dominate the top cross validation experiments applied to the *Indoor* versus *Outdoor* data set, while other parameters matter less.

Parameter	Value	Average Accuracy %
Text Span	first sentences of captions	81.83
	captions	78.13
	articles (including captions)	73.10
	articles (excluding captions)	71.44
Part-of-Speech	all words	76.41
	open-class words	76.35
	open-class words except proper nouns	75.62
Include Tag	yes	76.35
	no	75.90
Case Sensitive	no	76.13
	yes	76.12
Thresholds	none	77.04
	low	77.02
	medium	76.48
	high	73.97
Normalize	no	76.55
	yes	75.70
Density Estimation	small window	77.48
	medium window	77.23
	large window	77.15
	none	72.64

Table G.2: For some parameters, specific settings do much better than others over all cross validation experiments applied to the *Indoor* versus *Outdoor* data set; other parameters matter less.

- **Text Span**. Restricting analysis to the first sentences of captions accounts for the 67 top scoring experiments. First sentences clearly outperform full captions (based on the average overall accuracy of all cross validation experiments in which those settings are used), while text spans that include the entire article (with or without the caption) are far behind. So convincing is this result that in later experiments involving the *Indoor* versus *Outdoor* data set, discussed throughout this thesis, I assume from the beginning that this is the appropriate text span to use.

- **Restriction to specific grammatical categories**. Using only open-class words excluding proper nouns accounts for all of the top six, ten of the top 11, and 24 of the top 27 cross validation experiments. Interestingly, though, the average overall accuracy of all cross validation experiments using this setting is slightly lower than when either of the other two settings is used. In a case like this, using open-class words excluding proper nouns is probably the setting that should be chosen for this feature when training the system for this data set, but it is very important to use this setting with other settings that have led to the best cross validation experiments.

- **Disambiguation of words**. Disambiguation of words based on part-of-speech (i.e. including the part-of-speech tag as part of the word) is used in the top eight cross validation experiments, and the average overall accuracy of all cross validation experiments using disambiguation of words is slightly higher than that of the cross validation experiments that do not use disambiguation of words.

- **Case sensitivity**. The average overall accuracy of all cross validation experiments using versus not using case sensitivity are virtually identical, and both settings are used in some of the top cross validation experiments.

- **Thresholds for TF*IDF values**. Some threshold is used in each of the top 11 cross validation experiments, with medium thresholds being used in the top eight. Looking at average overall accuracy over all cross validation experiments with each setting, however, lack of thresholds does the best, followed very closely by low thresholds. Medium thresholds are not far behind, but high thresholds do considerably worse. Using medium thresholds is probably the setting that should be chosen for this feature when training the system for this data set, but it is very important to use this setting with other settings that have led to the best cross validation experiments.

- **Normalization of category vectors**. The average overall accuracy of all cross validation experiments that do not use normalization is slightly higher than that of those that do use normalization, and normalization is not used in the top two cross validation experiments, although both settings are used in some of the top cross validation experiments.

- **Density estimation**. Using density estimation accounts for the top 147 cross validation experiments. With almost every combination of the other parameters, adding density estimation seems to improve performance for this data set. The size of the window matters less, as all three sizes are used in some of the top experiments (although three of the top four use a small window, and this window size also slightly beats the other two over all cross validation experiments). Looking at the average overall accuracies of all cross validation experiments with each setting, all three window sizes do about the same (small windows do slightly better than medium windows which do slightly better than large windows), and experiments without density estimation are far behind. I show in Chapter 4 that density estimation does indeed have a very positive effect on the performance achieved by the system when applied to the test set of this data set.

G.3 Cross Validation Results for the Events Data Set

There are 2,560 possible combinations of the 7 optional parameters as described above for the Events data set (because of the two extra allowable settings tested for the part-of-speech parameter). Every one of these combinations has been tested using three-fold cross validation on the training set of this data set. As with the *Indoor* versus *Outdoor* data set, the settings of certain parameters turn out to be significantly more important than the settings of others.

Table G.3 shows the settings used for the optional parameters for the best ten cross validation results for the training set of the Events data set. These cross validation experiments all achieve an overall accuracy of approximately 84% or greater, with the single best cross validation experiment obtaining an overall accuracy of approximately 85.4%. Table G.4 shows the average overall accuracy of all cross validation experiments for each possible value of all these parameters. The effects of each of the various parameters varied in the cross validation experiments using the Events data set can be summarized as follows:

- **Text Span**. For the Events data set, using a concatenation of articles and captions accounts for the 24 top scoring experiments, and either articles alone or concatenations of articles with captions account for the top 142 experiments. Judging by the average overall accuracy of all cross validation experiments using each setting, concatenation of articles and captions clearly comes in first, with articles alone clearly beating full captions, and full captions clearly beating first sentences. This is almost the opposite of the result seen for the *Indoor* versus *Outdoor* data set, but it is not surprising due to the nature of the categories. The *Indoor* and *Outdoor* categories apply strictly to the image, and a description of the image (as provided by the first sentence

Rank	Text Span Used	Part-of-Speech Used	Include Tag	Case Sensitive	Thresholds	Normalize	Density Estimation
1	articles including captions	open-class words except proper nouns	yes	no	none	yes	none
2	articles including captions	open-class words except proper nouns	yes	yes	none	yes	none
3	articles including captions	open-class words except proper nouns	yes	yes	none	no	small window
4	articles including captions	open-class words except proper nouns	yes	yes	none	no	medium window
4	articles including captions	open-class words except proper nouns	yes	no	none	no	small window
4	articles including captions	open-class words except proper nouns	yes	no	none	no	medium window
7	articles including captions	nouns except proper nouns	yes	yes	none	yes	none
8	articles including captions	nouns except proper nouns	yes	no	no	yes	none
8	articles including captions	nouns except proper nouns	no	yes	none	yes	none
8	articles including captions	open-class words except proper nouns	yes	no	low	yes	none

Table G.3: Some parameters have specific settings dominate the top cross validation experiments applied to the Events data set, while other parameters matter less.

Parameter	Value	Average Accuracy %
Text Span	articles (including captions)	78.77
	articles (excluding captions)	76.59
	captions	74.45
	first sentences of captions	70.84
Part-of-Speech	open-class words	77.09
	nouns	76.31
	all words	76.02
	open-class words except proper nouns	74.12
	nouns except proper nouns	72.27
Include Tag	yes	75.26
	no	75.07
Case Sensitive	no	75.24
	yes	75.08
Thresholds	low	77.60
	none	77.46
	medium	76.64
	high	68.96
Normalize	yes	75.28
	no	75.05
Density Estimation	small window	76.11
	medium window	75.94
	large window	75.55
	none	73.05

Table G.4: For some parameters, specific settings do much better than others over all cross validation experiments applied to the Events data set; other parameters matter less.

of the caption) is clearly the most helpful text. The Events categories apply to the document as a whole, and define high-level, topical notions of what the entire story is about.

- **Restriction to specific grammatical categories**. Using only open-class words excluding proper nouns accounts for all of the top six and seven of the top ten cross validation experiments (with the other three using nouns except proper nouns). As with the *Indoor* versus *Outdoor* cross validation experiments, though, we see a different outcome when we look at the average overall accuracy of all cross validation experiments using each setting. The two settings that appear in the top experiments do the worst when averaged over all experiments in which they are used. Over all cross validation experiments, using open-class words does the best, followed by all words, and then followed by nouns; open-class words except proper nouns does better than nouns except proper nouns. This result is unusual, and can lead to confusion when choosing settings for the final training of the system; as with the *Indoor* versus *Outdoor* data set, I believe that choosing open-class words except proper nouns is the appropriate setting to choose, but it is extremely important to choose it along with other settings that lead to one of the best cross validation experiments.

- **Disambiguation of words**. Disambiguation of words based on part-of-speech (i.e. including the part-of-speech tag as part of the word) is used in the top seven cross validation experiments, and as with the *Indoor* versus *Outdoor* data set, the average overall accuracy of all cross validation experiments using disambiguation of words is slightly higher than that of the cross validation experiments that do not use disambiguation of words.

- **Case sensitivity**. The average overall accuracy of all cross validation experiments without case sensitivity is slightly higher than that of those with case sensitivity, and both settings are used in some of the top cross validation experiments.

- **Thresholds for TF*IDF values**. Unlike the results for the *Indoor* versus *Outdoor* data set, no threshold is used in the top seven and nine of the top 10 cross validation experiments. Looking at average overall accuracy over all cross validation experiments with each setting, however, low thresholds do the best, followed very closely by no thresholds. Medium thresholds are not far behind, but high thresholds do considerably worse.

- **Normalization of category vectors**. The average overall accuracy of all cross validation experiments that do use normalization is slightly higher than that of those that do not use normalization (the opposite of the result for the *Indoor* versus *Outdoor* cross validation experiments), and normalization is used in the top two cross validation experiments, although both settings are used in some of the top cross validation experiments.

- **Density estimation**. For me, the most surprising result of these cross validation experiments is that density estimation is not clearly helpful for this data set. Density estimation is not used in the top two and six out of the top ten cross validation experiments. This is quite different than the cross validation results for the *Indoor* versus *Outdoor* data set, for which density estimation accounts for the top 147 experiments. Interestingly, however, if we look at the average overall accuracies of all cross validation experiments using each setting, we see that density estimation with all three window sizes does better than lack of density estimation. Still, as explained previously, when choosing settings for the final training, it is best to choose those that lead to the best cross validation experiments. It is confirmed in Chapter 4 that using density estimation (with the parameters for which it does the best in cross validation experiments) leads to a slight degradation for the Events data set (a very different result than found for the other data sets examined in that chapter). It appears, then, that density estimation is helpful for the Events data set when used in conjunction with most other combinations of parameter settings, but when the other settings are just right, lack of density estimation does better.

G.4 Concluding Discussion of Cross Validation and Features

This appendix has demonstrated how cross validation can be used to test the importance of features and choose appropriate settings that are likely to achieve good performance. It is important to test all combinations of possible settings, as opposed to testing each feature on its own, because sometimes specific combinations do well together even though the individual settings might not do well on their own. The settings for parameters should be chosen based on their appearance in the top cross validation experiments. When the setting for a specific parameter seems ambiguous, it can also be helpful to look at average accuracies over all experiments using each setting for the parameter. The final combination of settings chosen should be one that leads to one of the top cross validation experiments.

In this appendix, I have examined seven features for my own Rocchio/TF*IDF system. Interestingly, the three of them that are probably the most common in the text categorization literature - case sensitivity, normalization, and thresholds to cutoff words with low weights - do not prove to be particularly important. Two other features involve tagging all words using a part-of-speech tagger, and both seem to be somewhat useful. Limiting text to words of particular grammatical categories proves to be important, although choosing the correct setting is tricky (the setting that is used in the top cross validation experiments is not the the one that gives the best accuracy over all cross validation experiments); using part-of-speech tags to disambiguate words also seems to help for both data sets. Another feature involves choosing the correct span of text to use for the experiments, and this proves to be especially crucial, although the appropriate setting clearly differs between data sets based on the nature of the categories. Finally, density estimation is extremely useful for one of the two data sets discussed in this appendix, but does not seem to be particularly useful for the other. Chapter 4 verifies this finding concerning density estimation; I show density estimation greatly improves the accuracy of the system when applied to the test set of the *Indoor* versus *Outdoor* data set, but mildly degrades the accuracy of the system when applied to the test set of the Events data set.

Appendix H

Density Estimation Probabilities

The density estimation approach towards text categorization has two main advantages. One is that the application of density estimation to the output of another system often improves the results of the system, as demonstrated in Chapter 4. The second main benefit is that predictions are assigned probabilistic confidence measures. These measures can potentially be used to combine the results of various systems. Although I have not yet used density estimation confidence measures for this purpose, I discuss a means of doing so in Section 8.3.3 and in Section 10.3.

In Chapter 4, I describe, in detail, the way in which density estimation estimates probability likelihoods that documents belong to categories. I will not reiterate that here. I also show, in that chapter, that density estimation often improves the overall accuracy of the system to which it is applied. However, I do not actually check to see if the probabilities are reasonably accurate. If these probabilities are to be used for other purposes, such as combining predictions of different systems, it must at least be true that higher probabilities are more likely to be correct, on average. This appendix tests that assumption for the *Indoor* versus *Outdoor* data set and for the Events data set.

We have seen in Chapter 4 that density estimation improves the overall accuracy of a state-of-the-art Rocchio/TF*IDF system (that I have also implemented) for the *Indoor* versus *Outdoor* data set from 80.7% to 86.1%, a significant gain. The overall accuracy of the system for the Events category, on the other hand, decreases from 87.1% to 84.9%, which is not significant according to a standard test. The question this appendix explores, however, is whether or not the probability estimates assigned by density estimation to its individual predictions are somewhat accurate and, therefore, potentially useful.

For the *Indoor* versus *Outdoor* data set, since there are only two categories, and density estimation selects the one with the highest probability assigned, the chosen category always has a probability of at least 50%. For this data set, I have defined three levels of confidence. *High confidence* refers to predictions for which the probability estimate for the chosen category is over (or equal to) 90%. *Medium confidence* refers to predictions for which the probability estimate is less than 90% but greater than (or equal to) 70%. *Low confidence* applies to the remaining predictions, which necessarily have a probability estimate of less than 70% but at least 50%.

Confidence Level	Number Correct	Number Incorrect	Overall Accuracy %
High	264	21	92.6
Medium	74	24	75.5
Low	45	17	72.6
Total	383	62	86.1

Table H.1: The density estimation probabilities assigned to predictions for the *Indoor* versus *Outdoor* data set seem to be indicative of the likelihood of the prediction being correct.

Table H.1 shows the results at various levels of confidence for the *Indoor* versus *Outdoor* data set. Sure enough, higher levels of confidence lead to higher overall accuracies. Most notably, the high confidence interval accounts for 285

(64.0%) of the 445 images in the test set, and for this subset of images, the system categorizes 92.6% of them correctly.

For the Events data set, since there are five categories, it is possible that the selected category (the one assigned the highest probability estimate) has a probability under 0.5. (The highest probability for this data set is necessarily greater than 0.2, since the probabilities must add up to 1.) For this data set, I have therefore defined four levels of confidence; The first three are the same as those used for the *Indoor* versus *Outdoor* data set, and the fourth is *very low confidence*, which refers to predictions for which the estimated probability is under 50%.

Confidence Level	Number Correct	Number Incorrect	Overall Accuracy %
High	284	17	94.4
Medium	54	14	79.4
Low	32	28	53.3
Very Low	6	8	42.9
Total	376	67	84.9

Table H.2: The density estimation probabilities assigned to predictions for the Events data set seem to be indicative of the likelihood of the prediction being correct.

Table H.2 shows the results at various levels of confidence for the Events data set. Once again, higher levels of confidence lead to higher overall accuracies; this time, the trend is even more noticeable than demonstrated in the previous table. In particular, the high confidence interval accounts for 301 (67.9%) of the 443 documents in the test set, and for this subset of documents, the system categorizes 94.4% of them correctly. Only 14 (3.2%) of the predictions are given very low confidence (probabilities under 50%), and only 42.9% of these predictions are correct.

This appendix clearly shows that the probability estimates assigned by density estimation do seem to indicate the likelihood that the predictions are, in fact, correct. As to the actual accuracy of the probabilities themselves (i.e. do they tend to be too high or too low, on average), this requires further testing. Based on Tables H.1 and H.2, the probabilities seem, at least, to be reasonably accurate; with only one exception (that being the low confidence interval for the *Indoor* versus *Outdoor* data set), the overall accuracy for each interval does fall into the range of the estimated probabilities covered by the interval. If these estimates are reasonably accurate in general, then the confidence measures provided by density estimation should be useful for combining predictions of various systems.

Appendix I

Burstiness

The BINS system described in Chapter 5 optionally allows additional features to be used as weights (in addition to or instead of IDF values). One such feature is burstiness, an idea introduced by Katz (Katz, 1996) and used by default in an earlier version of BINS described in (Sable and Church, 2001). Burstiness takes into account that some words are likely to appear many times in a document if they appear at all. The burstiness of a word, the way we define it, is either one or zero, depending on whether or not its average term frequency is greater than a specific cutoff. More formally, letting $Burstiness(w)$ be the burstiness of a word w, $TF(w)$ be the term frequency of the word (the number of occurrences of the word in a training corpus), $DF(w)$ be the document frequency of of the word (the number of documents in which the word occurs at least once in a corpus), and $IDF(w)$ be the inverse document frequency of the word as is first discussed in Section 2.4.2, we have:

$$Burstiness(word) = \begin{cases} 1 & \text{if } TF(w)/DF(w) > 1.83 - 0.048 \times IDF(w) \\ 0 & \text{otherwise} \end{cases} \tag{I.1}$$

All other things being equal, it is expected that bursty words (words with a burstiness of 1) are more meaningful (indicative of categories) than words that are not bursty.

Burstiness is intended to be used in conjunction with IDF; however, BINS also allows it to be used standalone if the user desires. (Category counts, as described in Section 5.3, are also automatically used as a binning feature in all experiments; this is required by the BINS system, as there is no reason to expect good performance without them.) Even if a feature such as burstiness is indicative, to some degree, of the probability that a word will appear in a future document of a category, this does not necessary mean that one should expect a performance gain by using it as an additional binning feature. Every new feature that bins uses to bin words makes each individual bin smaller (i.e. each bin consists of less words), and therefore the weights computed for each bin potentially might be less accurate. There is a performance gain only if the benefit of using an additional indicative feature outweighs the negative effect of making the bins smaller.

Tables I.1 and I.2 show the results of using burstiness as a feature for the *Indoor* versus *Outdoor* data set and the Events data set, respectively. For each data set, I have tried using burstiness by itself and also in conjunction with IDF. In addition to using this new feature when BINS always relies on bin weights, I have also measured the effect of the feature with both of the combinations of bin weights and Naive Bayes weights discussed in Section 5.7.

The result for both data sets seem inconclusive. For the *Indoor* versus *Outdoor* data set, using a combination of burstiness and IDF performs marginally better than IDF alone in two cases and equally in one case. Surprisingly, burstiness alone performs better than IDF alone in two cases (and equally in one case). This is surprising, since burstiness is always zero or one, and so there are much fewer bins using burstiness compared to IDF. Perhaps because the documents in the *Indoor* versus *Outdoor* data set are small (first sentences of captions), and therefore there are less total words in the training set of the corpus, using smaller bins is generally

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
BINS (always use bin)			
BINS (IDF weights)	85.8	75.1	90.1
BINS (burstiness)	87.6	77.7	91.4
BINS (IDF and burstiness)	86.1	75.4	90.3
BINS (COMBO #1)			
BINS (IDF weights)	86.1	76.2	90.2
BINS (burstiness)	86.5	76.4	90.6
BINS (IDF and burstiness)	86.3	76.6	90.3
BINS (COMBO #2)			
BINS (IDF weights)	87.2	78.0	91.0
BINS (burstiness)	87.2	77.3	91.1
BINS (IDF and burstiness)	87.2	78.0	91.0

Table I.1: The use of burstiness as a binning feature does not have a large effect on results for the *Indoor* versus *Outdoor* data set.

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
BINS (always use bin)						
BINS (IDF weights)	88.0	87.5	88.3	97.2	83.1	60.0
BINS (burstiness)	87.4	87.0	87.3	97.2	85.4	42.4
BINS (IDF and burstiness)	87.8	87.1	87.7	97.8	85.4	52.6
BINS (COMBO #1)						
BINS (IDF weights)	90.3	89.7	90.8	98.3	89.4	62.5
BINS (burstiness)	89.6	88.8	89.8	98.3	89.1	57.1
BINS (IDF and burstiness)	90.7	90.1	91.1	98.9	90.3	62.5
BINS (COMBO #2)						
BINS (IDF weights)	90.3	89.5	90.4	98.9	89.1	63.8
BINS (burstiness)	88.9	87.9	89.2	97.2	89.1	56.4
BINS (IDF and burstiness)	90.5	89.8	90.8	98.9	89.1	63.8

Table I.2: The use of burstiness as a binning feature does not have a large effect on results for the Events data set.

helpful. For the Events data set, using burstiness alone does not perform as well is IDF alone, as is generally expected, although the degradation of performance is never large. Using burstiness in addition to IDF leads to a marginal performance gain in two cases and a marginal degradation of performance in one case.

Results such as the ones described in this Appendix do not make it clear whether or not burstiness is generally be a good feature to use for binning. For these data sets, the effect is minimal. In (Umemura and Church, 2000), however, burstiness does improve performance for an information retrieval system that relies on bins, and their analysis shows that the feature can be helpful. In any case, my BINS system does not use the feature by default, but it does allow users to compute and use the feature if they so specify.

Appendix J

Shared Scaled Category Likelihoods

The BINS system described in Chapter 5 optionally allows additional features to be used as weights (in addition to or instead of IDF values). One such feature, burstiness, has been discussed in Appendix J. In this chapter, I describe an additional optional feature that can be used to bin words. The feature is, to the best of my knowledge, a novel mathematical measure, which I call the *shared scaled category likelihood* (SSCL) of a word. This measure estimates the likelihood that two documents sharing a word in common also belong to the same category. The measure is meant as an alternative to IDF, one that specifically takes the task of text categorization into consideration; my BINS system also allows the two measures to be used in combination.

J.1 Description of SSCL

The method used to estimate SSCLs assumes that the category distribution in the training set is somewhat indicative of what the category distribution will be in future documents. This assumption is being made for the corpus as a whole and also for individual words. So, if a particular category c occurs with frequency f_1 in the training set, it is assumed that the same category will occur with approximately frequency f_1 in future documents. Furthermore, considering only the documents in that training set that contain a particular word w, if a particular category c occurs with frequency f_2 within this set of documents, I am assuming that in all future documents containing w, the same category will occur with approximately frequency f_2. Clearly, this may not truly be a good estimate for words that occur infrequently or just once, for instance, and perhaps these estimates could be improved by using concepts such as binning and deleted interpolation, but these are the assumptions for now.

The method also assumes that categories are mutually exclusive and that every document is assigned exactly one category. A corpus allowing multiple (or zero) categories per document can be converted to a corpus with one category per document by replacing every possible set of categories with a single category representing that set (i.e. use each possible power set of the original categories as a single category). Of course, this increases the number of possible categories from n to 2^n, but in practice far fewer than 2^n actually occur, and only those categories that occur at least once in the training set need to be considered.

Along the way to estimating the SSCL for every possible word w, I first estimate the probability that two future documents that both contain a word w do, in fact, share the same category. To do this, I compute the probability that two documents selected *with replacement* from the set of training documents that contain the word belong to the same category. The reason for assuming selection with replacement, as opposed to without replacement, when calculating the probabilities can be made apparent by considering a simple example. Suppose that a particular word w occurs in exactly two documents in the training set, and that these documents have categories c_1 and c_2. The assumptions above thus lead us to assume that approximately half of the future documents with word w are likely to belong to category c_1 and approximately half are likely to belong to category c_2. (Expressed more formally, $P(c_1|w) = 0.5$ and $P(c_2|w) = 0.5$.) We are not making any assumptions about the number of total future documents that we will see, and there is no reason to conclude that seeing one future document with word w belonging to category c_1 makes it less likely that the next document with word w belongs to category c_1. So, it is fair to conclude that if two future documents both contain the word w, there is approximately a $1/4$ probability that both documents belong to category c_1, a $1/4$ probability that both documents belong to category c_2, and a $1/2$ probability that one document belongs to c_1 and the other belongs to c_2. More specific to the point, there is approximately a $1/2$ probability that the two documents share the same category and a $1/2$ probability that they do not. This matches the probabilities that are calculated based on the training set if selection with replacement is considered. However, if selection without replacement is considered, the method would calculate that there is 0 probability that two future documents both containing the word w share the same category (and a probability of 1 that they do not), since, in the training set, there is only one pair of documents that both contain w and they do not share the same category. This, clearly, would not be a wise estimate if the category distribution of documents containing the word w is assumed to be similar to that already observed in the training set.

A first pass through the training set computes the following statistics:

- N - the number of training documents.
- $size(c)$ - the number of documents belonging to category c.
- $DF(w)$ - the number of documents with word w.
- $count(c, w)$ - the number of documents with word w belonging to category c.

Then, the estimated probability that two documents share the same category given that they share the word w becomes:

$$P(same|w) = \frac{\sum_{c \in C} count(c, w)^2}{DF(w)^2} \tag{J.1}$$

At this point, I could simply use the logs of these values as word weights. Of course, since probabilities are all less than or equal to one, all weights would then be negative or zero. Instead, I first take the ratio of each probability to the global probability, based on the entire training corpus, that two randomly selected documents (with replacement) belong to the same category:

$$P(same) = \frac{\sum_{c \in C} size(c)^2}{N^2} \tag{J.2}$$

I then compute word weights as follows:

$$\lambda(w) = \log_2 \frac{P(same|w)}{P(same)} \tag{J.3}$$

Dividing by the global probability of two documents sharing the same category offsets all word weights by the same magnitude (since the weight is a log of the ratio). The advantage is that term weights are positive for words that increase the probability of two documents sharing the same category, while term weights are negative for words that decrease the probability of two documents sharing the same category. A system that uses these weights might decide to ignore negative term weights (if it is decided that it is unfair to penalize two documents for sharing any words in common).

Although it might seem odd that any word should ever indicate that documents sharing the word are significantly less likely to share the same category than two documents that don't contain the word, this can be the case for certain corpora. In fact, with an older version of BINS that could also handle binary categories, I have tried using this approach to compute word weights for the Reuters corpus (first described in Section 2.8), and I have seen that many common words have negative term weights. The term weight for “the”, for example, comes out to approximately -0.723. This can be explained by noting that the presence of the word “the” in two documents makes both documents less likely to be in the Reuters *earn* category, the largest category containing mostly numerical documents.

J.2 Partial Justification of SSCL

As a partial justification of this approach, it can easily be shown that if the training set contains exactly one sample document of every category, this approach for estimating term weights for words is equivalent to using inverse document frequencies (IDFs) for all words based on the same training set:

$$\begin{aligned}
\lambda(w) &= log_2 \frac{P(same|w)}{P(same)} \\
&= \log_2 \frac{\frac{\sum_{c \in C} count(c,w)^2}{DF(w)^2}}{\frac{\sum_{c \in C} size(c)^2}{N^2}} \\
&= \log_2 \frac{\frac{1^2 * DF(w)}{DF(w)^2}}{\frac{1^2 * N}{N^2}} \\
&= \log_2 \frac{1/DF(w)}{1/N} \\
&= \log_2 \frac{N}{DF(w)} = IDF(w)
\end{aligned}$$

This does not hold true if the category distribution in the training set is different, and it is easy to see that different words with the same document frequencies (and thus the same IDFs) have different term weights depending on the category distribution of the documents they belong to. For example, consider two words, w_1 and w_2 that each occur in ten documents in the training set. If w_1 occurs in ten documents of the same category but w_2 occurs in ten documents with ten different categories, it can easily be seen from the formulas above that the estimated probability that two documents containing w_1 belong to the same category is ten times higher than that for w_2, and therefore the final term weight for the w_1 is higher by a magnitude of $log_2 10$. This seems to fit intuition; that is to say, for the purposes of text categorization, it seems intuitive that w_1 should be considered more indicative than w_2.

One problem with the method as it is currently implemented is that if two words occur in the training set with the same category distributions, they are assigned identical term weights regardless of how common each word is. This is clearly not ideal for words that occur very infrequently. For example, right now, a word that occurs only one time in the training set (and, therefore, in only one category) is assigned the maximum possible weight, the same as that of a word that occurs many times in the training set but still in only one category. However, IDF weights have a similar problem; the highest weights are being assigned to the words with the least evidence. This problem with SSCL can potentially be countered by using concepts such as binning and deleted interpolation, but that is not something I have yet tried. For now, the method seems to provide reasonable weights for individual words.

J.3 Results and Evaluation of SSCL as a Binning Feature

SSCL weights are intended to be used instead of IDF weights, as I have designed the measure to be an improved but similar measure specifically for the purpose of text categorization; however, BINS also allows the two weights to be used in combination if the user desires. (Category counts, as described in Section 5.3, are also automatically used as a binning feature in all experiments; this is required by the BINS system, as there is no reason to expect good performance without them.) Using a combination of features is not always a good idea even if all the features are indicative, to some degree, of the probability that a word will appear in the future document of a category. Every new feature that BINS uses to bin words makes each individual bin smaller (i.e. each bin consists of less words), and therefore the weights computed for each bin potentially might be less accurate. There is a performance gain only if the benefit of using an additional indicative feature outweighs the negative effect of making the bins smaller.

Tables J.1 and J.2 show the results of using SSCLs as a feature for the *Indoor* versus *Outdoor* data set and the Events data set, respectively. For each data set, I have tried using SSCL by itself (as intended) and also in conjunction with IDF. In addition to using this new feature when BINS always relies on bin weights, I have also measured the effect of the feature with both of the combinations of bin weights and Naive Bayes weights discussed in Section 5.7.

The results based on these experiments are inconclusive. For the *Indoor* versus *Outdoor* data set, using SSCL alone does as well as using IDF alone for two cases, and slightly worse for a third case. Using the two features together performs worse than IDF alone in all three cases; perhaps because the documents for this data set are small (only first sentences of captions are used), there may not be enough data to support the smaller bins. For the Events data set, SSCL alone does marginally worse than IDF in two cases and equally well for a third case. This time, however, the combination of IDF and SSCL performs better than either weight alone in all three cases. Although I have designed SSCL to be used as a replacement for IDF as opposed to being used in conjunction with it, it is possible that the two weights express complementary information, and perhaps they should be combined when there is enough data to support it.

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
BINS (always use bin)			
BINS (IDF weights)	85.8	75.1	90.1
BINS (SSCL weights)	85.8	74.5	90.2
BINS (IDF and SSCL)	84.0	71.4	88.9
BINS (COMBO #1)			
BINS (IDF weights)	86.1	76.2	90.2
BINS (SSCL weights)	86.1	75.4	90.3
BINS (IDF and SSCL)	85.4	74.9	89.7
BINS (COMBO #2)			
BINS (IDF weights)	87.2	78.0	91.0
BINS (SSCL weights)	86.3	75.7	90.5
BINS (IDF and SSCL)	86.3	75.9	90.4

Table J.1: The use of SSCL as a binning feature does not have a large effect on results for the *Indoor* versus *Outdoor* data set.

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
BINS (always use bin)						
BINS (IDF weights)	88.0	87.5	88.3	97.2	83.1	60.0
BINS (SSCL weights)	87.6	87.3	87.6	97.2	85.4	45.7
BINS (IDF and SSCL)	88.5	88.0	88.6	97.8	85.4	54.1
BINS (COMBO #1)						
BINS (IDF weights)	90.3	89.7	90.8	98.3	89.4	62.5
BINS (SSCL weights)	90.1	89.4	89.8	98.9	89.1	63.8
BINS (IDF and SSCL)	90.7	89.9	90.8	98.9	89.1	66.7
BINS (COMBO #2)						
BINS (IDF weights)	90.3	89.5	90.4	98.9	89.1	63.8
BINS (SSCL weights)	90.3	89.5	89.8	98.8	89.1	66.7
BINS (IDF and SSCL)	90.5	89.9	91.1	98.3	89.1	60.5

Table J.2: The use of SSCL and IDF in combination may be helpful for the Events data set.

J.4 Concluding Discussion of SSCL

The results of the experiments so far do not make it clear whether or not shared scale category likelihoods would generally perform as well as inverse document frequencies when used for text categorization tasks. For both of the data sets used in this appendix, the two weights perform about equally as well as each other when either is used alone. For one of the two data sets, there is a small improvement when the two weights are used together, but for the other data set, there is a small degradation in performance. I feel that these results do not merit using SSCL by default for my BINS system, as opposed to IDF, which is vastly used in the text categorization literature; however, the system does allow a user to specify that these weights should be computed and used, either alone or in combination with IDF.

Appendix K

IDF: More Data versus Better Data

When computing IDFs (inverse document frequencies, as defined in Section 2.4.2), it is customary to compute these values based on the training set, but another possibility is to compute the values based on some other, much larger set of documents. On the one hand, using the training set ensures that the documents have a similar style and format to documents that appear in the test set (or will appear later); in other words, training data tends to be representative of the data that will be seen in the future. On the other hand, sometimes more data is better data. See, for example, recent papers by Banko and Brill (Banko and Brill, 2001b; Banko and Brill, 2001a) which argue that much larger training sets may be more important than better algorithms for certain natural language tasks (they provide evidence that this is so for a task they call confusion set disambiguation).

In this appendix, I compare using the training set to compute IDFs to using approximately one million AP News documents. (Category counts, as described in Section 5.3, are also automatically used as a binning feature in all experiments; this is required by the BINS system, as there is no reason to expect good performance without them.) This comparison is made for the *Indoor* versus *Outdoor* data set (first described in Section 3.1.2.2) and also for the Events data set (first described in Section 3.1.2.3). Each of these data sets contains close to, but less than, 1,000 training examples. AP articles are approximately the same size, on average, as articles in my data set, so using the values based on the AP corpus is basing the values on at least 1,000 times the evidence, and the ratio is even larger for the *Indoor* versus *Outdoor* data set, since that corpus uses only first sentences of captions. BINS can not compute IDFs based on the AP corpus by itself, but in addition to allowing the user to compute and use certain optional features for binning (see Appendices I and J for two examples), BINS also allows a user to provide his or her own weights by providing a file that maps each word to a weight. I have created such a file specifying IDF weights computed using the AP corpus.

Tables K.1 and K.2 compare the results using each type of IDF value as a binning feature for the *Indoor* versus *Outdoor* data set and the Events data set, respectively. In addition to examining the effect on performance when BINS always relies on bin weights, I have also measured the effect with both of the combinations of bin weights and Naive Bayes weights discussed in Section 5.7. The results are inconclusive. In each case, when bin weights are always used, the use of AP IDFs leads to improved performance, although not by a huge margin. (This matched the results we saw for an older version of BINS, as reported in (Sable and Church, 2001), which did not allow bin weights to be combined with Naive Bayes weights.) When bins are used in conjunction with Naive Bayes weights, the use of AP IDFs leads to a clear degradation in performance for the *Indoor* versus *Outdoor* data set, but the effect on performance for the Events data set is marginal (with the direction depending on the combination of weights used).

It is possible that the approach of BINS for using user-provided weights (such as the AP weights in this case) is not optimal. The way BINS is set up, if a user

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
BINS (always use bin)			
BINS (IDF weights)	85.8	75.1	90.1
BINS (IDF from AP corpus)	86.5	76.2	90.6
BINS (COMBO #1)			
BINS (IDF weights)	86.1	76.2	90.2
BINS (IDF from AP corpus)	85.2	74.8	89.5
BINS (COMBO #2)			
BINS (IDF weights)	87.2	78.0	91.0
BINS (IDF from AP corpus)	85.8	75.7	90.0

Table K.1: Using IDFs based on the AP corpus instead of those computed based on the training data of the *Indoor* versus *Outdoor* data set improves performance when bins are always used, but degrades performance when bins are used in combination with Naive Bayes.

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
BINS (always use bin)						
BINS (IDF weights)	88.0	87.5	88.3	97.2	83.1	60.0
BINS (IDF from AP corpus)	88.5	86.8	88.4	97.8	87.9	63.6
BINS (COMBO #1)						
BINS (IDF weights)	90.3	89.7	90.8	98.3	89.4	62.5
BINS (IDF from AP corpus)	90.5	89.7	91.4	98.9	89.1	62.7
BINS (COMBO #2)						
BINS (IDF weights)	90.3	89.5	90.4	98.9	89.1	63.8
BINS (IDF from AP corpus)	90.1	89.7	91.1	97.8	89.1	65.3

Table K.2: Using IDFs based on the AP corpus instead of those computed based on the training data of the Events data set does not have a large effect on performance.

provides his or her own weights, any word that is not assigned a weight is ignored. For the *Indoor* versus *Outdoor* data set, there are 4,656 distinct words in the training set, and 4,524 (97.2%) of them have AP weights (i.e. these words appeared at least once in the corpus of approximately one million AP news documents used to compute the AP IDFs). This means that 132 (2.8%) of the distinct words in the training set are ignored for this data set. For the Events data set, there are 21,912 distinct words in the training set, and 21,099 (96.3%) of them have AP weights. This means that 813 (3.7%) of the distinct words in the training set are ignored for this corpus. It is uncertain if these words would play an important role in predicting categories if they were somehow used. Potential methods of using these words would be to assign all of the unknown words the same weight (and therefore bin this set of words separately using only category counts), or by using IDFs computed based on the training set (the default) for these words. I have not explored these possibilities.

The BINS system, by default, computes IDF weights based on the training set, which is standard for text categorization systems. The results in this appendix are certainly not decisive enough to merit otherwise. Even if there were a clear gain from using the AP IDFs, this would not necessarily be a good choice. Loading the AP weights is slower than computing the weights based on the training corpus. More importantly, I do not want users of BINS to be required to have access to outside information for training purposes, and it is unclear if I would have permission to provide, along with the system, IDFs that I have computed using the AP corpus. Additionally, there are cases where the use of AP IDFs leads to serious degradation in performance. For example, see the discussion in (Sable and Church, 2001) concerning the use of an alternate (and older) version of BINS applied to the Reuters data set (this alternate version of BINS can handle binary categories). The AP IDFs cause a problem because of certain eccentricities with the Reuters data set. For example, the use of the abbreviations “mln” for “million” and “pct” for “percentage” are both very common in Reuters, and while these words have been assigned AP IDFs (i.e. they each occur at least once in a million documents), they are quite rare in the AP corpus and are assigned IDF values that are much higher than appropriate for Reuters; this throws off results.

Appendix L

Using Only Category Counts to Bin Words

As described in Section 5.3, my BINS system uses two features by default to group words into bins; namely, the "category count" and the quantized (truncated) inverse document frequency (IDF) of each word is used. (The IDF measure is first described in Section 2.4.2.) In other words, if two words share the same category count in a category and the same IDF, they are placed into the same bin for that category. (BINS also allows other optional binning features, two of which have been described in Appendix I and Appendix J.) Section 5.4.3 justifies the use of IDF as a binning feature by presenting statistical evidence that IDF is indicative of the future probability of seeing a word in a category, even when the category count of the word in the category is held constant. The end of that section, however, explains that this does not necessarily mean that performance improves by using this binning feature. Every new binning feature makes bins smaller, and therefore potentially less accurate; performance only improves if the benefit of using an additional indicative feature outweighs the negative effect of smaller bins. BINS therefore allows a user to specify that only category counts should be used to bin words, and this appendix explores the effect of that option for the *Indoor* versus *Outdoor* data set and for the Events data set.

Tables L.1 and L.2 compare the results using only category counts as a binning feature against the results using category counts in combination with IDF for the *Indoor* versus *Outdoor* data set and the Events data set, respectively. In addition to examining the effect on performance when BINS always relies on bin weights, I have also compared the two options using both of the combinations of bin weights and Naive Bayes weights discussed in Section 5.7. The results are inconclusive. For the Events data set, the use of IDF as an additional binning feature improves results in all three cases, although not by a huge margin. For the *Indoor* versus *Outdoor* data set, the results go down in two cases and stay the same in the third case. The results go down the most in the case when only bin weights are used. This could be because the training set of the *Indoor* versus *Outdoor* data set contains much fewer words (since only first sentences of captions are used), and so the negative effect of making bins smaller by the introduction of the additional feature is more prominent. Still, based on the justification of IDF as a binning feature presented in Section 5.4.3, and also the vast use of the feature in the text categorization literature, I have decided that BINS will use the feature to bin words by default.

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
BINS (always use bin)			
BINS (IDF weights)	85.8	75.1	90.1
BINS (only category counts)	87.6	77.7	91.4
BINS (COMBO #1)			
BINS (IDF weights)	86.1	76.2	90.2
BINS (only category counts)	86.5	76.4	90.6
BINS (COMBO #2)			
BINS (IDF weights)	87.2	78.0	91.0
BINS (only category counts)	87.2	77.3	91.1

Table L.1: Using IDF in addition to category counts actually degrades performance for the *Indoor* versus *Outdoor* data set in two out of three cases.

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
BINS (always use bin)						
BINS (IDF weights)	88.0	87.5	88.3	97.2	83.1	60.0
BINS (only category counts)	87.4	87.0	87.7	97.2	84.1	42.4
BINS (COMBO #1)						
BINS (IDF weights)	90.3	89.7	90.8	98.3	89.4	62.5
BINS (only category counts)	89.6	88.8	89.8	98.3	89.1	57.1
BINS (COMBO #2)						
BINS (IDF weights)	90.3	89.5	90.4	98.9	89.1	63.8
BINS (only category counts)	88.9	87.6	89.1	97.2	89.1	57.9

Table L.2: Using IDF in addition to category counts improves performance for the Events data set.

Appendix M

Summary of Optional Binning Features

Tables M.1 and M.2 summarize the results of the experiments discussed in Appendices I, J, K, and L for the *Indoor* versus *Outdoor* data set and the Events data set, respectively. The first line of each chart shows the results when only category counts are used to bin words. This feature, explained in Section 5.3, is always required, and there is no reason to expect that decent results can be achieved without it. The remaining lines of the charts show the results when other features are used in addition to category counts.

The second line of each chart represents the default configuration of BINS; in this case, the IDFs of words, calculated based on the first half of the training set, are also used for binning. This feature is described in Section 2.4.2. As explained in Appendix L, the addition of this binning feature improves results for the Events data set but seems to lead to a slight degradation in performance for the *Indoor* versus *Outdoor* data set. Although these results are inconclusive, I have decided that BINS will use the feature to bin words by default; this decision has been based on the justification of IDF as a binning feature presented in Section 5.4.3, and also

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
BINS (always use bin)			
BINS (only category counts)	87.6	77.7	91.4
BINS (default IDF weights)	85.8	75.1	90.1
BINS (IDF from AP corpus)	86.5	76.2	90.6
BINS (SSCL weights)	85.8	74.5	90.2
BINS (IDF and SSCL)	84.0	71.4	88.9
BINS (burstiness)	87.6	77.7	91.4
BINS (IDF and burstiness)	86.1	75.4	90.3
BINS (COMBO #1)			
BINS (only category counts)	86.5	76.4	90.6
BINS (default IDF weights)	86.1	76.2	90.2
BINS (IDF from AP corpus)	85.2	74.8	89.5
BINS (SSCL weights)	86.1	75.4	90.3
BINS (IDF and SSCL)	85.4	74.9	89.7
BINS (burstiness)	86.5	76.4	90.6
BINS (IDF and burstiness)	86.3	76.6	90.3
BINS (COMBO #2)			
BINS (only category counts)	87.2	77.3	91.1
BINS (default IDF weights)	87.2	78.0	91.0
BINS (IDF from AP corpus)	85.8	75.7	90.0
BINS (SSCL weights)	86.3	75.7	90.5
BINS (IDF and SSCL)	86.3	75.9	90.4
BINS (burstiness)	87.2	77.3	91.1
BINS (IDF and burstiness)	87.2	78.0	91.0

Table M.1: This table summarizes the results for the *Indoor* versus *Outdoor* data set of the optional features examined in the previous four appendices.

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
BINS (always use bin)						
BINS (only category counts)	87.4	87.0	87.7	97.2	84.1	42.4
BINS (default IDF weights)	88.0	87.5	88.3	97.2	83.1	60.0
BINS (IDF from AP corpus)	88.5	86.8	88.4	97.8	87.9	63.6
BINS (SSCL weights)	87.6	87.3	87.6	97.2	85.4	45.7
BINS (IDF and SSCL)	88.5	88.0	88.6	97.8	85.4	54.1
BINS (burstiness)	87.4	87.0	87.3	97.2	85.4	42.4
BINS (IDF and burstiness)	87.8	87.1	87.7	97.8	85.4	52.6
BINS (COMBO #1)						
BINS (only category counts)	89.6	88.8	89.8	98.3	89.1	57.1
BINS (default IDF weights)	90.3	89.7	90.8	98.3	89.4	62.5
BINS (IDF from AP corpus)	90.5	89.7	91.4	98.9	89.1	62.7
BINS (SSCL weights)	90.1	89.4	89.8	98.9	89.1	63.8
BINS (IDF and SSCL)	90.7	89.9	90.8	98.9	89.1	66.7
BINS (burstiness)	89.6	88.8	89.8	98.3	89.1	57.1
BINS (IDF and burstiness)	90.7	90.1	91.1	98.9	90.3	62.5
BINS (COMBO #2)						
BINS (only category counts)	88.9	87.6	89.1	97.2	89.1	57.9
BINS (default IDF weights)	90.3	89.5	90.4	98.9	89.1	63.8
BINS (IDF from AP corpus)	90.1	89.7	91.1	97.8	89.1	65.3
BINS (SSCL weights)	90.3	89.5	89.8	98.8	89.1	66.7
BINS (IDF and SSCL)	90.5	89.9	91.1	98.3	89.1	60.5
BINS (burstiness)	88.9	87.9	89.2	97.2	89.1	56.4
BINS (IDF and burstiness)	90.5	89.8	90.8	98.9	89.1	63.8

Table M.2: This table summarizes the results for the Events data set of the optional features examined in the previous four appendices.

the vast use of the feature in the text categorization literature. The third line of each chart presents results when an alternative version of IDF, based on the analysis of approximately one million AP News documents, is used instead of the default IDF. Although this is significantly more data than available in the training set of my own corpus, the data is less representative. A comparison of the results using the two types of IDF is inconclusive; at the end of Appendix K, I explain why I have decided to use IDFs based on the training set by default.

The remaining lines of each chart present the results using optional binning features, either in addition to IDF or instead of IDF. One of these binning features is burstiness, described in Appendix I, and the other is a measure I call shared scaled category likelihood (SSCL), described in Appendix J. Once again, results are inconclusive. At the end of Appendices I and J, I explain why I have decided that BINS will not use these features by default.

Appendix N

Using Unlabeled Data with BINS

N.1 Why Use Unlabeled Data?

As described in Section 2.5, in order to train a text categorization system, manually labeled examples of the various categories need to be provided. This usually entails obtaining human volunteers who agree to examine various documents and label them accordingly. Once a text categorization system is implemented, this manual collection process is often the most time consuming (and the most annoying) phase of moving to a new set of categories.

If unlabeled data could somehow be used to improve categorization, on the other hand, that is much simpler to obtain. Generally, vast amounts of unlabeled documents are available. This has inspired two methods of using just a small handful of *seeds* (i.e. labeled examples) along with a large amount of unlabeled examples. One of these methods is *bootstrapping* (Yarowksy, 1995) and the other is *co-training* (Blum and Mitchell, 1998) (which is actually a type of bootstrapping). A recent paper by Abney (2002) compares the two algorithms.

It may some unintuitive that unlabeled data should be useful at all. Why would any training document be helpful in making predictions for future documents if you don't know the training document's category? One way to think of it is like this: If a system is trained on only the labeled examples, and then it is tested on the unlabeled data, there will be certain documents that are part of the unlabeled data for which the system might be very confident in its prediction. If we consider only those unlabeled documents for which the confidence is above some very high cutoff, a very high majority of the predictions for these documents should be accurate. Adding these specific unlabeled documents to the training set (as examples of their predicted categories) might be beneficial. The reason is that these confident decisions may have been made based on certain evidence in these documents, and the documents may contain additional indicative evidence of categories that is not obvious from the original training data. This procedure (training based on labeled examples, testing the unlabeled data, and adding documents with confident predictions to the training set) could potentially be repeated iteratively until things converge or until some other stopping criteria is met. The hope is that the final training set is better than the original training set. (This algorithm is similar to although not exactly the same as what bootstrapping and co-training are doing. I do not explain those algorithms in detail here, but the papers I have cited are good sources.)

This appendix describes experiments with BINS using unlabeled data in addition to labeled data for the *Indoor* versus *Outdoor* data set (first described in Section 3.1.2.2) and the Events data set (first described in Section 3.1.2.3). I will first explain how unlabeled documents are used to create a new binning feature for words indicating the number of unlabeled documents containing a word that are confidently predicted in each possible category. It turns out that the addition of this binning feature has not improved performance for these data sets. At the end of the appendix, I will discuss possible reasons for this, and I will explain why I still believe there is promise in the approach.

N.2 Using Unlabeled Category Counts as a Feature for BINS

The algorithms mentioned in the previous section allow documents that are originally not labeled to be added to a training set as examples of categories for which they are confidently predicted. One potential problem, however, is that the labels of these documents are likely not quite as accurate as those of manually labeled documents. It might therefore be better to weight these added documents less in some way, but most systems do not provide a way to do this (or they provide a method that is ad-hoc). It turns out that a bin-based system, on the other hand, provides a very natural way to weight documents that are originally unlabeled less than manually labeled documents.

Remember that BINS creates a separate set of bins for each category, as described in Section 5.3. One of the features used to group words into bins, described in that same section, is the category count of each word; i.e. the number of documents of the specified category in the first half of the training set that contain the word. In fact, the category count of a word for a category is the only feature that BINS requires for binning (and one of two features used be default). Now, in addition to manually labeled training documents, let's say we also have unlabeled documents, some of which have very confident predictions from BINS (after it is trained on the labeled data). For each word/category pair, we can now define a second count, which is the number of unlabeled documents that have been confidently predicted to belong to the category. We will call this the word's *unlabeled category count* for the specified category.

Let's say we are dealing with the Events data set (originally defined in Section 3.1.2.3). Further, some given word w_1 with an IDF of x does not occur in any *Disaster* documents in the first half of the training set, nor does it occur in any unlabeled documents that have been confidently predicted to belong to the *Disaster* category. Another word w_2 that also has an IDF of x does not occur in any *Disaster* documents in the first half of the training set, but it does occur in ten unlabeled documents that have been confidently predicted to belong to the *Disaster* category. It is arguable, at the very least, that the *Disaster* term weight for w_2 should be higher than the *Disaster* term weight for w_1. This can be accomplished be using unlabeled category counts as an additional binning feature, so that w_1 and w_2 will be placed in different bins. The beauty of this is that the method used by BINS to calculate term weights for bins (described in Section 5.4.1) will empirically estimate weights for each bin, thereby providing a natural way of determining the importance of this new binning feature. If it turns out that unlabeled category counts are not really important (i.e. they are not indicative of categories), then it is expected that bins corresponding to different values of this feature but sharing other features in common will be assigned very similar weights. If, on the other hand, this new feature does turn out to be important, then it is expected that bins corresponding to different values of this feature, all else being equal, will be assigned different weights, most likely with a progression towards higher weights as the unlabeled category count for a specified category increases.

N.3 Determining High Confidence Predictions

As explained in Section 5.5, BINS generates a sum for every category. These sums represent log likelihoods of a document similar to the one being examined occurring in each possible category (under certain assumptions such as the independence assumption). The category with the highest sum is the one that is predicted. One possible way to measure the confidence of this prediction is to take note of the difference between the sum generated for the chosen category and the sum generated for the category that comes in second place.

In order to determine whether or not a prediction meets some particular standard of confidence, a cutoff must be chosen such that any difference above this cutoff is considered good enough. This cutoff can be selected once for all categories, or a separate cutoff can be selected for each category (i.e. when a specific category is predicted, the difference between the sum for this category and the sum for the second place category will be compared to the cutoff that corresponds to the chosen category). I have used the later approach, since it is not clear that the sums for all categories are on an equal scale.

Of course, it is not valid to consider the labels of the test set when determining cutoffs for each category. These cutoffs must be based on the training set. One solution is to perform cross validation experiments within the training set, and to choose cutoffs that maximize the F_1 measure for each category. Remember, though, as described in Section 2.7, the F_1 measure weights precision and recall equally. In this particular case, it might be better to weight precision more than recall, since having an inaccurately labeled document in the training set is possibly worse than leaving out an accurately labeled document. As described in Section 2.7, the more general form of the F_1 measure is the F_β measure, and lower values of β weight precision higher than recall. By lowering the value of β, we are therefore weighting precision higher as compared to recall, which will cause cutoffs for categories to increase; in other words, less of the unlabeled documents will be added to the training set, but it is expected that a higher percentage of those documents that are added will have accurate labels. In my experiments using unlabeled data for both the *Indoor* versus *Outdoor* data set and the Events data set, I have tried using β values of 1.0, 0.5, 0.33333, and 0.1 (therefore weighting precision equal to recall, twice as high as recall, approximately three times as high as recall, and ten times as high as recall). The next two sections demonstrate how the value of β affects the number of unlabeled documents used to compute unlabeled category counts.

N.4 Adding Unlabeled Data to the Indoor versus Outdoor Data Set

Remember that the training set for the *Indoor* versus *Outdoor* data set, first described in Section 3.1.2.2, consists of 894 documents (captioned images) including 621 *Outdoor* documents and 273 *Indoor* documents. In addition, I also have available 2,684 unlabeled captioned images. If BINS is trained on its original training set, and then applied to these unlabeled documents, it predicts that 2,042 of the documents are *Outdoor* documents and that 642 of the documents are *Indoor* documents. It is possible to use all of these predictions to obtain unlabeled category counts for BINS. Or, a cutoff can be determined for each category based on cross validation experiments within the training set, as explained in Section N.3. I have computed cutoffs that maximize F_β values, using values of 1.0, 0.5, 0.33333, and 0.1 for β.

β	# of Additions to Each Category	
	Indoor	Outdoor
none	642	2042
1.0	618	2042
0.5	539	1964
0.33333	355	1568
0.1	71	1479

Table N.1: The number of unlabeled documents used to obtain unlabeled category counts for the *Indoor* and *Outdoor* categories depends on the value of β that is used to determine cutoffs for each category.

Table N.1 shows the number of unlabeled documents confidently predicted for each category, where a confident prediction is one that surpasses the cutoff chosen to maximize the specified F_β measure. If β is set to 1.0 or 0.5, almost all of the unlabeled documents are used. If β is set to 0.33333, the number of confident *Indoor* predictions is cut almost in half, while the number of confident *Outdoor* predictions is cut by about 25%. If β is set to 0.1, the number of confident *Indoor* predictions is cut be almost 90%, while the number of confident *Outdoor* predictions does not fall much from the 0.33333 cutoff.

N.5 Adding Unlabeled Data to the Events Data Set

The training set for the Events data set, first described in Section 3.1.2.3, consists of 885 documents (including full news articles) including 282 *Struggle* documents, 243 *Politics* documents, 207 *Disaster* documents, 100 *Crime* documents, and 53 *Other* documents. In addition, I also have available 8,156 unlabeled articles. If BINS is trained on its original training set, and then applied to these unlabeled documents, it predicts that 2,070 are *Struggle* documents, 4,393 are *Politics* documents, 993 are *Disaster* documents, 468 are *Crime* documents, and 232 are *Other* documents. To determine cutoffs for confident predictions, I have performed three-fold cross validation experiments within the training set and computed cutoffs that maximize the same F_β values used for the *Indoor* versus *Outdoor* data set.

Table N.2 shows the number of unlabeled documents confidently predicted for each category in the Events data set, where a confident prediction is one that surpasses the cutoff chosen to maximize the specified F_β measure. It should not be too surprising that the affect is quite different for different categories. Note that most of the predictions for the *Disaster* category are very confident, as few are filtered even when β is set to 0.1 (thus weighting precision ten times as important as recall). This makes sense, as we have seen throughout the thesis that all text based systems tested perform extremely well for this category. Few predictions for the

β	# of Additions to Each Category				
	Struggle	Politics	Disaster	Crime	Other
none	2070	4393	993	468	232
1.0	2070	3285	934	468	232
0.5	955	2288	840	468	144
0.33333	955	2288	840	468	65
0.1	283	300	840	326	17

Table N.2: The number of unlabeled documents used to obtain unlabeled category counts for each category of the Events data set depends on the value of β that is used to determine cutoffs for each category.

Other category, on the other hand can withstand such a strict confidence cutoff, and this also makes sense because we have seen throughout this thesis that all systems perform poorly for this category. Somewhat surprising is that the *Politics* category dominates the predictions (over half of the unlabeled data is predicted to belong to this category), but less than 7% of these predictions are confident enough to withstand the strictest cutoff (over half withstand the cutoff one level up).

Figures N.1 and N.2 both show how precision, recall, F_1 and $F_{1/3}$ vary with different confident cutoffs for the *Struggle* category based on three-fold cross validation experiments on the training data of the Events data set. The cutoff indicates the score that needs to be beat for a prediction (or non-prediction, if the score is negative) of the *Struggle* category to stand. Not surprisingly, as we raise this cutoff, precision for the category increases (since we are being more discriminating), but recall falls (since we are placing fewer documents in the category). The first figure shows the entire range of confidences achieved; in other words, the most confident prediction for the *Struggle* category scores it just under 300 above the second place category, whereas the furthest that *Struggle* ever placed behind the first place category is nearly 600. The second figure highlights the key range of confidence cutoffs from zero to 100.

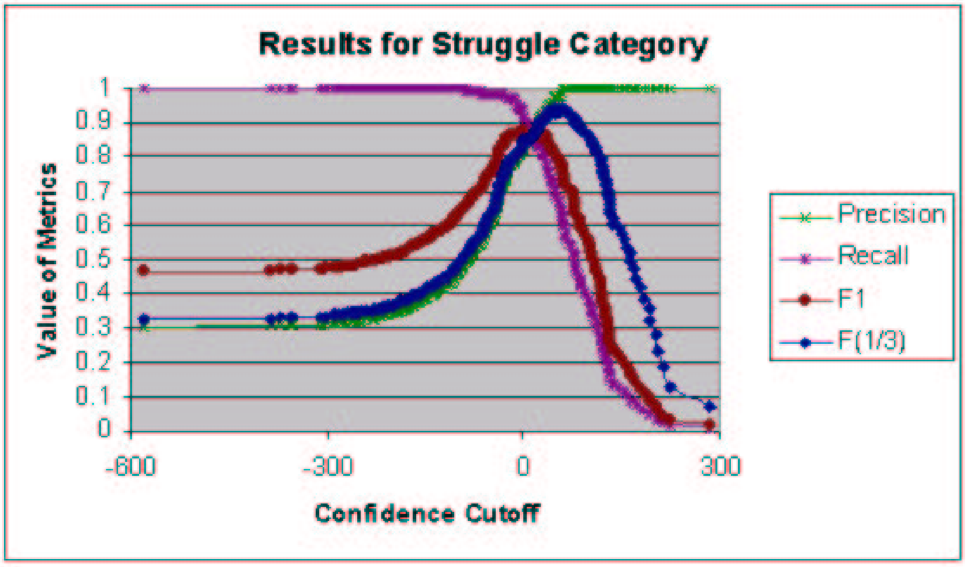


Figure N.1: As the confidence cutoff increases, precision increases but recall falls; depending the value of β, the F_β measure is maximized at different locations.

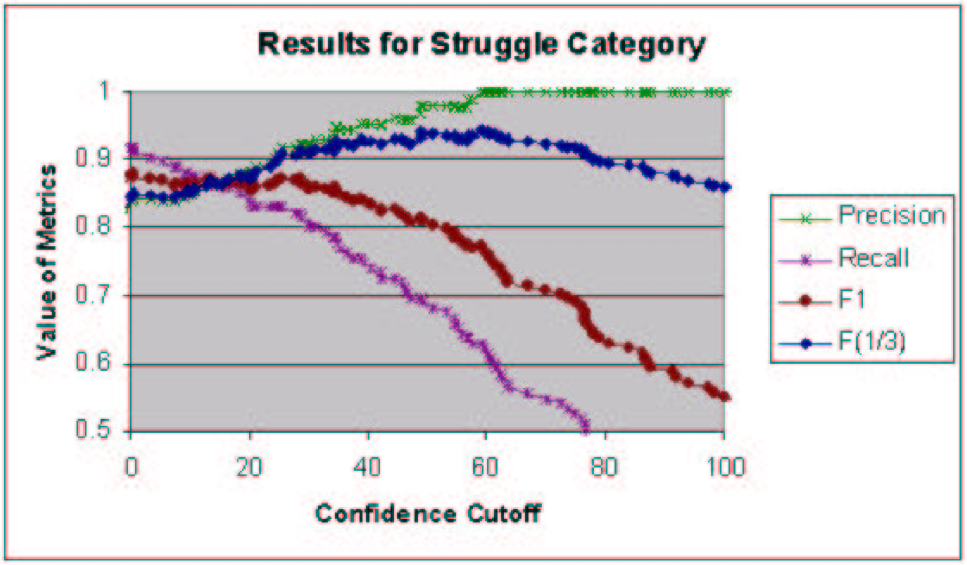


Figure N.2: The F_1 measure is maximized at a cutoff of approximately zero, whereas the $F_{1/3}$ measure is maximized at a cutoff of approximately 60.

Note that the F_1 measures maximizes at a cutoff of almost exactly zero, at which point the precision is slightly under 85% and the recall is a bit over 90%. In other words, to maximize the F_1 measure for this category, it makes sense to consider all predictions for this category to be confident. This matches the information from Table N.2, which shows that with a β of 1.0, all predictions for the *Struggle* category are considered confident. You can also see from that table that this is not true for every category. On the other hand, the $F_{1/3}$ measure, which weights precision three times as important as recall, is maximized at a cutoff of approximately 60. Using this cutoff, the precision for the *Struggle* category based on the cross validation experiments is a perfect 100%, but the recall has fallen to slightly over 60%. You can see from Table N.2 that with this cutoff, less than half of the *Struggle* predictions on the unlabeled data are considered confident.

N.6 Are Unlabeled Category Counts Indicative?

Once we decide cutoffs for categories and use them to obtain confident predictions for certain unlabeled documents, we can use these confident predictions to obtain unlabeled category counts for words, as defined in Section N.2. BINS provides mechanisms for computing per category confidence cutoffs that maximize a specified F_β, for determining which unlabeled documents have predictions that exceed this confidence, and for using these unlabeled documents to produce unlabeled category counts that are used as an additional feature. Once the BINS system has been trained, the trained model produced based on the training data and the unlabeled data can be examined to check if the term weights seem to suggest that unlabeled category counts are helpful. In other words, all other things being equal, do term weights for bins that differ only in the unlabeled category count feature seem to vary in a somewhat predictable manner? If unlabeled category counts are indicative, it is expected that higher values of this feature, when all else is constant, would lead to greater term weights and therefore increased likelihood of a word from a particular bin appearing in a document of a particular category.

Figures N.3 and N.4 both show the influence of the unlabeled category count on the lambda of a bin (which is the term weight for the bin, corresponding to the likelihood of seeing a word from the bin show up in a document of some specific category). The first figure examines a trained model using unlabeled data with confident predictions based on a β of 0.5. It looks only at bins corresponding to normal category counts of zero and an IDF of eight. A line is plotted for each of the four major categories in the Events data set. For each of these categories, values of lambdas have been calculated holding everything constant but varying the unlabeled category count from zero through nine (there is no bin for at least one category with an unlabeled category count of ten). The second figure examines a trained model using unlabeled data with confident predictions based on a β of 1.0. It looks only at bins corresponding to normal category counts of zero, but this time separate plots are made for IDFs ranging from five through eight, and the category is held constant as the *Disaster* category. For each of the IDFs examined, values of lambdas have been calculated holding everything else constant but varying the unlabeled category count from zero through ten.

It is easy to see that there is a definite trend, and this trend is the expected one if unlabeled category counts are indicative of the likelihood of a word showing up in a document of some specified category. Remember from Section 5.4.1 that these term weights are log likelihoods using a base of two. Judging from these plots, it looks like words that never show up in the first half of the training set in a particular category but do show up in several unlabeled documents that are confidently predicted to belong to the same category are eight to 16 times as likely to appear in a new document of this category compared to words that never show up in the category in the first half of the training set and also only show up in very

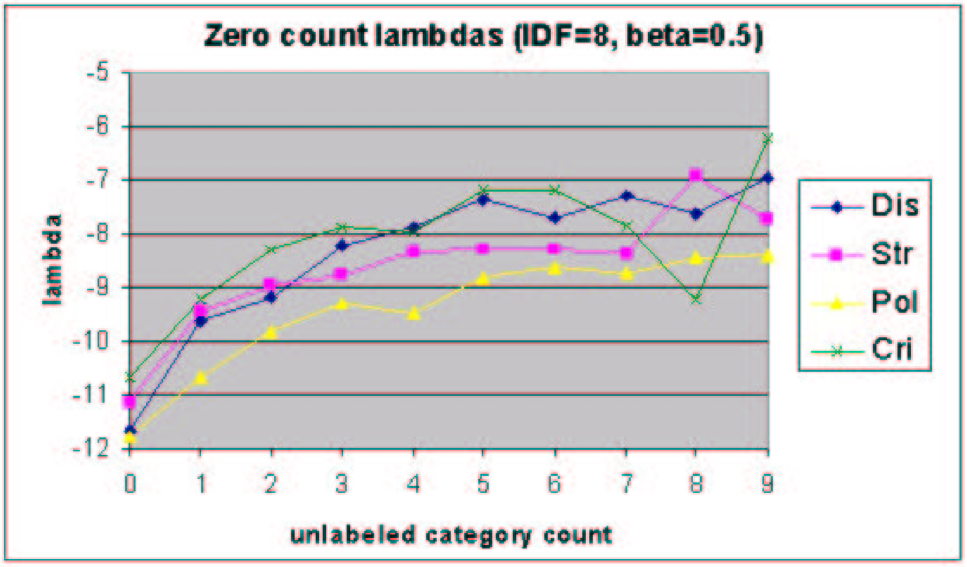


Figure N.3: The trend of seeing lambdas increase with increasing unlabeled category counts holds for all categories, but the lines are somewhat jagged.

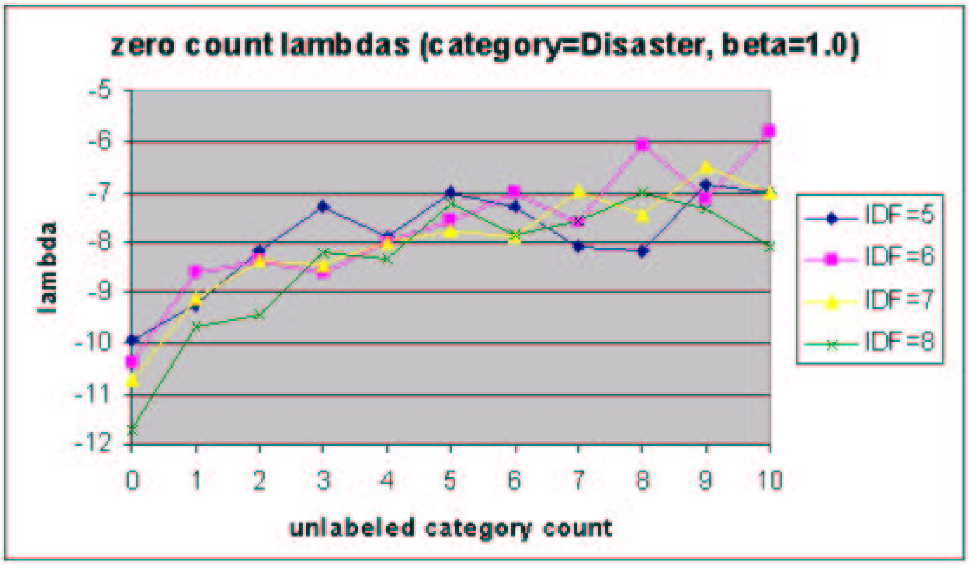


Figure N.4: As the unlabeled category count increases, holding everything else constant, the likelihood of seeing a word in a document of a specified category tends to increase.

few or no documents that are confidently predicted to belong to the category. This provides a justification for using this binning feature similar to that provided for IDF in Section 5.4.3.

However, notice that none of the lines are consistently increasing. Every plot has at least one instance of falling as the unlabeled category count increases. Some of the lines are actually quite jagged. This brings up the negative effect of adding an additional binning feature. Every added feature may provide more indicative information about words, but it also makes the bins smaller. Every bin that exists with the default configuration of BINS is divided into potentially many smaller bins based on the various possible unlabeled category counts. Thus, bins may become less accurate, especially those that have scarce evidence. This could explain the jaggedness of these plots, and it might be responsible for the negative results that will now be discussed.

N.7 Results and Evaluation of Experiments with Unlabeled Data

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
BINS with unlabeled data			
BINS (only labeled data)	85.8	75.1	90.1
BINS (maximize F_1)	84.5	72.5	89.2
BINS (maximize $F_{1/2}$)	84.5	72.5	89.2
BINS (maximize $F_{1/3}$)	84.0	71.7	88.9
BINS (maximize $F_{1/10}$)	83.6	72.2	88.4

Table N.3: The use of unlabeled category counts based on unlabeled data as an additional binning feature degrades performance for the *Indoor* versus *Outdoor* data set.

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
BINS with unlabeled data						
BINS (only labeled data)	88.0	87.5	88.3	97.2	83.1	60.0
BINS (maximize F_1)	86.9	85.8	87.7	96.6	84.4	48.6
BINS (maximize $F_{1/2}$)	87.4	86.2	88.3	96.7	84.4	52.6
BINS (maximize $F_{1/3}$)	87.4	85.4	89.3	96.7	84.1	57.8
BINS (maximize $F_{1/10}$)	87.1	86.1	88.4	97.2	81.4	53.7

Table N.4: The use of unlabeled category counts based on unlabeled data as an additional binning feature degrades performance for the Events data set.

Tables N.3 and N.4 show the performance of BINS using unlabeled data for the *Indoor* versus *Outdoor* data set and the Events data set, respectively. For both data sets, the unlabeled data is used to compute unlabeled category counts for words as an additional binning feature. Confident predictions are those that surpasses cutoffs computed to maximize a specific F_β measure based on three-fold cross validation experiments using the labeled training data. As you can see, the overall accuracy for each experiment using unlabeled data is less than that when BINS uses only labeled data (although only by one or two percent). Although unlabeled category counts do seem to be indicative of categories, as explained in Section N.6, this increased information apparently does not compensate for the loss of accuracy due to smaller bins.

N.8 Concluding Discussion of Unlabeled Data

The use of unlabeled data to improve performance of systems has received a lot of attention lately due to the vast amount of unlabeled data available and the difficulty of obtaining labeled data. Most methods of using unlabeled data involve iteratively adding unlabeled documents with confident predictions to the training set until some stopping point is reached, starting with just a few seed examples.

The goal is usually to achieve performance that is on par with performance that can be achieved by a manually labeled training set.

The experiments described in this chapter are different. Starting with a full training set achieving good performance for two tasks, I have attempted to improve performance further by incorporating unlabeled data. I have pointed out that most other methods using unlabeled data treat the unlabeled documents as equal to the labeled documents once it is decided that they will be used. BINS provides a mechanism of using the unlabeled documents but weighting them less than normal documents. By computing unlabeled category counts as an additional binning feature, term weights can be empirically estimated that take unlabeled documents into account without giving them too much weight. As shown in this appendix, these unlabeled category counts do seem to be indicative of the likelihood of a word from a particular bin appearing in a document of some specific category. Unfortunately, this extra information does not compensate for the loss of accuracy due to smaller bins, and the performance of BINS slightly degrades when unlabeled data is used for the *Indoor* versus *Outdoor* data set or the Events data set.

One question that arises is whether or not the size of the training sets used for these experiments without unlabeled data are already large enough such that more data is not particularly helpful. There is no reason to expect that additional unlabeled data will be as good as the same amount of additional labeled data, and it certainly won't be any better. Of course, if I had additional labeled data, I would have already been using it, so this is not really testable. What is testable, though, is using less training data. Appendix O tests what happens when I start with only one tenth of the available training data for each of the data sets used in this appendix, and add training data until the entire training set is used. I show that performance still seems to be increasing with training set size, although I may have reached a level of diminishing returns. It is not clear whether or not significant improvement can be achieved by increasing the training set size further. Since the benefit of additional unlabeled data is not expected to be as high as the benefit of additional labeled data, perhaps the task I have discussed in this appendix is too difficult. Maybe a more achievable goal would be to show that unlabeled data can be used in additional to a much smaller training set to achieve performance comparable to that achieved with a full training set. This would be more in line with other research involving unlabeled data. I leave this for future work.

Appendix O

The Effect of Training Set Size on Performance

Section 2.5 first describes the general machine learning paradigm for text categorization. Systems learn how to make predictions for future documents based on provided examples. These examples are usually in the form of manually labeled documents, and obtaining these examples can be the most expensive part of moving on to a new set of categories. Typically, the more training data is obtained, the better machine learning systems will perform, assuming that all of the data is equally accurate. In fact, recent papers by Banko and Brill (Banko and Brill, 2001b; Banko and Brill, 2001a) argue that much larger training sets may be more important than better algorithms for certain natural language tasks (they provide evidence that this is so for a task they call confusion set disambiguation).

This appendix studies the effect of training set size on the performance of my BINS system for the *Indoor* versus *Outdoor* data set (first described in Section 3.1.2.2) and the Events data set (first described in Section 3.1.2.3). I can not perform this study by adding to the training set that I usually use for these data sets; if I had more labeled data, I would have been using it the whole time. However, it is easy to use less training data. Therefore, the approach I have taken for each data set is to arbitrarily divide the training set into ten pieces of approximately equal size, and then to add one piece at a time to the training set to see how the system performs after each piece. The experiments discussed in this appendix run BINS with all of its default settings.

There are at least two reasons that this study is interesting. For one, in its own right, it is interesting to see how much of an effect the size of the training set has on performance. Since obtaining manual labels is often the most expensive part of moving to a new set of categories, it is helpful to know how many labels it pays to collect. In addition, these experiments have partially been conducted to help explain the results of Appendix N. In that appendix, I show that the use of unlabeled data, using a methodology explained in that appendix, does not help, and event slightly degrades, the performance of BINS for the same two data sets used in this appendix. There is no reason to believe, however, that unlabeled data should be as helpful as manually labeled data, and so if it turns out that a point of diminishing returns has already been reached (i.e. the training set has already reached a size such that more data is not particularly helpful), this could mean that I have attempted something quite difficult in Appendix N.

Table O.1 shows the results adding each new piece of the training set for both the *Indoor* versus *Outdoor* data set and the Events data set. Figure O.1 graphically depicts the same information. Not surprisingly, the performance of BINS improves as the size of the training set is increases, although there are a few small anomalies. It does appear, though, that the rate of improvement is diminishing, as the lines appear near flat by the time that the entire training set is being used. It is probable that more improvement can be achieved by increasing the size of the training set further, but the level of such improvement is hard to guess.

I would expect that the improvement adding another 10% or 20% to either

% of Training Data Used	Overall Accuracy %	
	Indoor versus Outdoor	Events
10	72.8	59.8
20	75.5	75.4
30	74.6	80.4
40	80.9	83.3
50	83.4	85.8
60	82.7	86.9
70	84.3	86.2
80	84.7	86.2
90	85.2	88.7
100	85.8	88.0

Table O.1: Not surprisingly, the performance of BINS improves as the size of the training set increases, although the rate of improvement is diminishing.

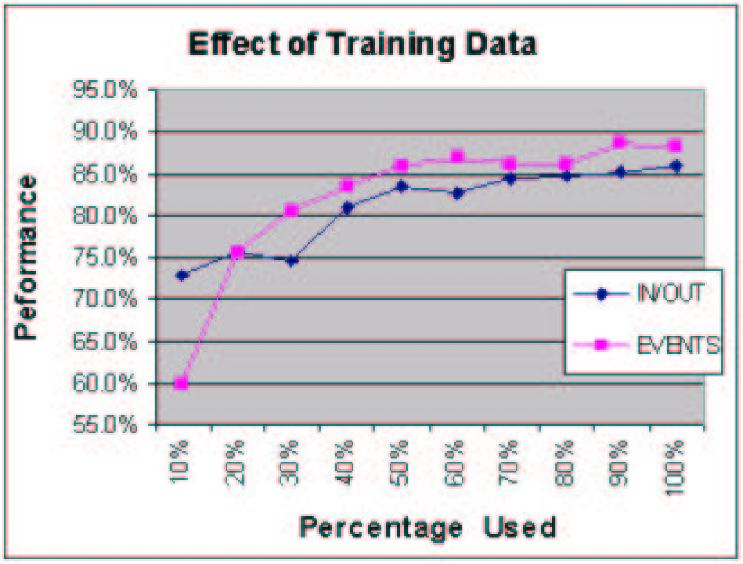


Figure O.1: The performance of BINS improves as the training set size increases, but the lines seem to be flattening by the time the entire training set is used.

training set would be small. Perhaps if one could increase the size of the training set by 100% of 1000%, the improvement might be much greater. Another possibility, however, is that each performance is asymptotically approaching a level to which it is already close; this especially makes sense for the *Indoor* versus *Outdoor* data set, which likely has an upper bound of approximately 87.6% (as explained in Section 3.1.2.2, this is the accuracy of humans who predict a category after viewing only the captions of the images), and the system is already achieving close to this. In fact, there has to be some upper bound, which the system eventually asymptotically approaches; at best, this can be 100%, but it is likely something less (I see no reason to believe that a system will achieve perfect performance given infinite data).

Appendix P

An Automatic Training Set for Newsblaster

When it first came time to apply text categorization to Newsblaster clusters, as described in Section 9.1.3, it was speculated by others working on the project that it may be possible to create a training set for the system automatically. At the time, there were five categories for news stories; they were the same as the six categories that exist now except that the *Science/Technology* category did not yet exist. The idea was that by copying articles from specific directories of sources that would almost surely belong to certain categories (e.g. directories containing articles placed in the *World News* or *Sports* section of other sites), we could find many training documents for each of our categories without having to manually label anything. Having already been aware of the importance of accurate and representative training sets (see the discussion in Section 2.5), I was skeptical, and it turns out that my skepticism was correct. This appendix describes the attempt to create an automatic training set for Newsblaster and compares results to those obtained from a manually labeled training set.

The automatic training set for Newsblaster has been created not by me, but by Sergey Sigelman, a member of our research group and the Newsblaster team. Sergey has collected articles from various news sites that, as previously described, are very likely to belong to specific Newsblaster categories. All together, Sergey's training set contains 7,326 documents including 1,269 *U.S. News* articles, 604 *World News* articles, 1,604 *Finance* articles, 693 *Entertainment* articles, and 3,156 *Sports* articles. Already, you may see a problem; these numbers are not really indicative of how frequently each category occurs on a normal day, they are simply indicative of how easy it is to find articles that are likely to belong to each category without reading them. That being said, I do not believe that this problem accounts for the results of this appendix; in my personal research experiences, I have noticed that most approaches are not affected very heavily by *a-priori* probabilities of categories.

In order to compare the automatic training set to a manually created training set, I have personally examined 2,000 articles from old Newsblaster runs and I have manually labeled them according to my own opinions. I have allowed myself to label an article as ambiguous (and therefore not include it in the training set) if I could not decide on the appropriate category, although I have tried to keep this to a minimum; this is how I have labeled 108 of the articles. My manually created training set thus consists of the remaining 1,892 documents, including 668 *U.S. News* articles, 811 *World News* articles, 223 *Finance* articles, 65 *Entertainment* articles, and 125 *Sports* articles. Note that this is a much smaller training set than Sergey's automatically created training set, but I will show that this is one instance in which the quality of the training data matters more than the quantity of the training data.

In order to compare the use of the two training sets, it would not have been fair to simply take a random portion of my manually labeled data and use it as a test set. This would have given my training set an advantage, as the training data would have clearly been more similar to the test data (since each set would have contained articles about the same stories). Therefore, I waited several weeks, and then manually labeled a single days worth of Newsblaster articles. At that time, Newsblaster did not download as many articles per day as it does now; there were 371 articles on that particular day. I have labeled these articles, once again allowing myself to prune ambiguous articles but trying to keep this to a minimum; I have therefore pruned only 15 ambiguous articles. The test set thus consists of the remaining 356 documents, including 83 *U.S. News* articles, 142 *World News* articles, 42 *Finance* articles, 13 *Entertainment* articles, and 76 *Sports* articles.

System	Overall Accuracy %		
	Automatic Training Set	Manual Training Set	Difference
My systems			
BINS	49.2	93.3	+44.1
Rainbow systems			
Naive Bayes	68.8	96.6	+27.8
Rocchio/TF*IDF	80.6	97.5	+16.9
K-Nearest Neighbor	50.6	87.1	+36.5
Probabilistic Indexing	91.3	94.4	+3.1
Support Vector Machines	71.1	96.1	+25.0
Maximum Entropy	67.7	94.7	+27.0

Table P.1: All tested systems perform better with the manual training set than they do with the automatic training set, and for all but one of the systems it makes a huge difference.

Table P.1 shows the results when systems are trained using either the automatic training set or the manual training set, and tested on a single days worth of Newsblaster articles. As you can see, all systems perform better using the manual training set even though it is much smaller. All but one of the systems are affected to an extreme degree. My BINS system is affected the most; performance degrades a whopping 44.1% when the automatic training set is used. Rainbow's Probabilistic Indexing system is affected the least by far, with performance degrading only

3.1%. The average performance difference is a change of 25.8% in overall accuracy. The fact that all of the systems do better (most of them by a huge margin) using the manual training set does not surprise me; this is what I expected. What does surprise me is that the level to which systems are affected varies so highly from one system to another. If you look throughout this thesis, for most text categorization tasks, all of the text based systems perform relatively close to one another; they are all, after all, using bag of words approaches and the same features to make their decisions. This task, however, when using the automatically created training set, has by far the largest performance range I have seen, spanning from 49.2% to 91.3%.

The degradation of performance using the automatically created training set is probably not caused by inaccurate labels for these documents. I have inspected the training set, and it seems to me that the overwhelming majority of them are correctly labeled; in fact, I have seen no document in that training set that is obviously in the wrong category. The problem is that this training set is not representative of future data, and the reason is subtle. Each category has representative articles in the training set from certain sources but not others. For example, The automatic training set for the *World News* category contains articles from only ABCNews, CNN, and Reuters; this is because these are the only sites for which Sergey has found directories containing articles that clearly belong in this category. Other categories contain representative articles in the training set from other sources.

There happen to be 319 articles in the training set, for example, taken from the Washington Post news site. 317 of these 319 articles are in the training set for the *U.S. News* category, only two belong to the training set for the *Finance* category, and the other three categories have no representative articles from this source at all. This is an example of a major problem, because the Washington Post often refers to itself (or to a "Washington Post reporter") in its articles, and systems might therefore learn that the word "post" is very indicative of the *U.S. News* category (unless this word happens to also turn up commonly in other categories for some other reason). However, in the test set, 120 of the 356 test articles happen to come from the Washington Post, and only 28 of these 120 articles actually belong to the *U.S. News* category. Of course, there may be other words in these articles that overcompensate for the misinformation from the word "post", and different systems using different approaches will weight the evidence differently; but if errors like this one occur commonly (and I believe they do), then this is likely what accounts for (at least part of) the degradation in performance using the automatic training set.

One might point out that for this particular data set, BINS seems to underperform most of the Rainbow systems, and thus one might question the decision to use it for Newsblaster categorization. However, I am confident with that decision for several reasons. First of all, the performance of BINS with the automatic training set is irrelevant. Newsblaster is using a manually constructed training set; after starting with a combination of the manual training set and test set described in this appendix, the training set has improved over time (as explained in Section 9.2). Second, the performance using the manual training set, as described in this appendix, is on par with the performance of the other systems, all of which do well. According to Table P.1, several of the Rainbow systems still seem to beat BINS by a few percentage points, even when using the manual training set. However, there happen to be several articles in the test set about the about Senate hearings involving the Enron scandal. At the time, I personally considered these to belong to the *Finance* category, although I realized they could arguable be placed in the *U.S. News* category as well; now, looking back, I even feel this might be more appropriate. It turns out that BINS does place these articles in the *U.S. News* category, and therefore gets penalized, whereas most of the other systems (all those with a higher overall accuracy) place them in *Finance*; this is responsible for the entire difference. In addition, as I have shown throughout the thesis, BINS tends to do well for tasks similar to this one (e.g. it has the best performance of all systems tested for the Events categories). Finally, I often check the daily categorization results of the Newsblaster system, and it is clearly performing very well. Generally, mistakes only occur when an unusual story appears, the likes of which has not been seen before; this would cause trouble for any system, and I have already implemented a means to correct such errors over time (as explained in Section 9.2).