USING BACKGROUND KNOWLEDGE TO IMPROVE TEXT CLASSIFICATION

BY SARAH ZELIKOVITZ

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Haym Hirsh

and approved by

New Brunswick, New Jersey

October, 2002

ABSTRACT OF THE DISSERTATION

Using Background Knowledge to Improve Text Classification

by Sarah Zelikovitz

Dissertation Director: Haym Hirsh

Automatic text categorizers use a corpus of labeled textual strings or documents to assign the correct label to previously unseen strings or documents. Often the given set of labeled examples, or "training set", is insufficient to solve this problem. Our approach to this problem has been to incorporate readily available information into the learning process to allow for the creation of more accurate classifiers. We term this additional information "background knowledge."

We provide a framework for the incorporation of background knowledge into three distinct text classification learners. In the first approach we show that background knowledge can be used as a set of unlabeled examples in a generative model for text classification. Using the methodology of other researchers that treat the classes of unlabeled examples as missing values, we show that although this background knowledge may be of a different form and type than the training and test sets, it can still be quite useful. Secondly, we view the text classification task as one of information integration using WHIRL, a tool that combines database functionalities with techniques from the information-retrieval literature. We treat the labeled data, test set and background knowledge as three separate databases and use the background knowledge as a bridge to connect elements from the training set to the test set. In this way, training examples are related to a test example in the context of the background knowledge. Lastly, we use Latent Semantic Indexing in conjunction with background knowledge. In this case background knowledge is used with the labeled examples to create a new space in which the training and test examples are redescribed. This allows the system to incorporate information from the background knowledge in the similarity comparisons between training and test examples.

Acknowledgements

Earning a Ph.D. is a project that spans many years, and that cannot be done without the constant guidance and assistance from many others. At this time, I must acknowledge all those who have helped me to arrive at this point.

My first thanks go to my adviser, Dr. Haym Hirsh, who creates a wonderful, stimulating academic environment for his students. He provided direction to my research with its twists and turns along the way, and was instrumental in introducing me to many researchers in my field. Aside from outstanding academic knowledge and research skills, I have learned much about writing, teaching, and advising from him as well.

I would also like to thank my three other thesis committee members, Dr. Lou Steinberg, Dr. Craig Nevill-Manning and Dr. Tom Mitchell for their insightful comments and ideas. Dr. William Cohen deserves special mention for his helpfulness in his area of expertise, giving of his time and work freely to help a struggling graduate student. Dr. Michael Littman deserves thanks for that as well. I must express appreciation to Dr. Kamal Nigam for his help with my experiments that were most related to his thesis work.

The Machine Learning Research Group, directed by Dr. Haym Hirsh, was the source of many of my ideas and experiments throughout my years at Rutgers. I wish to thank Arunava Banerjee, Chumki Basu, Brian Davison, Daniel Kudenko, David Loewenstern, Sofus Macskassy, Chris Mesterharm, Khaled Rasheed and Gary Weiss. Your input was invaluable, and your encouragement and quick replies to emergency e-mails have made the last few years easier.

Dr. Kate Goelz deserves a special thank you for enhancing my years at Rutgers with her inimitable educational methods, and by being a true friend. I hope that my future supervisors are as professional and understanding as you are.

I also want to thank Val Rolfe for the warmth that she gives to the department and for always being available during times of joy or crisis.

Lastly, and most of all, I would like to thank my family, without whom I could not have begun, continued, or completed the studies toward my degree. All the favors – both big and small – that you have done for me over these years will never be forgotten. I can only hope that we are able to reach many more milestones together.

Portions of Chapter 4 appeared previously in the paper ”Improving Short-Text Classification using Unlabeled Background Knowledge to Assess Document Similarity”, by Sarah Zelikovitz and Haym Hirsh, which is in the Proceedings of the Seventeenth International Conference on Machine Learning, 2000, and a part of Chapter 5 appeared previously in the paper ”Using LSI for Text Classification in the Presence of Background Text”, by Sarah Zelikovitz and Haym Hirsh, in the Proceedings of the Tenth International Conference on Information and Knowledge Management, 2001. I wish to acknowledge that much of this research has been funded by NASA Grant NCC2-1259.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Text Categorization

The abundance of digital information that is available has made the organization of that information into a complex and vitally important task. Automated categorization of text documents plays a crucial role in the ability of many applications to sort, direct, classify, and provide the proper documents in a timely and correct manner. With the growing use of digital devices and the fast growth of the number of pages on the World Wide Web, text categorization is a key component in managing information.

For the purposes of this thesis, text categorization (or alternatively, text classification) can be defined simply as follows: Given a set of documents D, and a set of m classes (or labels) C, define a function F that will assign a value from the set of C to each document in D. For example, D might consist of the set of all classified advertisements in a newspaper, and C would therefore be the set of headings in the classified section of that same newspaper. If an element of D (taken from http://classifieds.dailyrecond.com/cv3/dailyrecord) is:

CARPENTRY All home improvements including decks, siding,
wood restoration, waterprfg, roofs, kitch/bath remod
& repairs, etc. 973-328-6876

then it would be assigned the value of *Business & Service* from the set C.

In many text classification problems, multiple classes may actually be assigned to one specific document. For example, if the set D consists of the set of titles of papers on mathematics, one paper might belong to the three classes: *number theory, classical analysis,* and *numerical analysis*. In that case, the definition above on text classification would still apply, if a separate classification problem was designed for each class, where C in each new problem would simply consist of the set $< t, f >$, t signifying that the document belongs in the class and f that it does not.

Problems in text classification vary widely in many aspects. Some documents to be classified are very short text strings such as titles of Web pages, titles of papers or articles, or even simply names of companies. Others may consist of entire Web pages, email contents, or news articles, sometimes spanning many pages. There are document classification problems with many possible classes that possibly even overlap. Other problems may have only two or three classes with clear and unique definitions for the classes.

Some examples of the current uses and research interests of automatic text categorization are:

- Topic Detection and Tracking: Streams of data such as news stories or broadcasts are used to detect new topics, or to track old topics in the news [1]. This allows interesting stories to be flagged, or stories on a topic of interest to be sent to a user with a notice of high priority (http://www.ldc.upenn.edu/Catalog/LDC98T25.html).

- Personalizing approaches to providing information: This area includes personalizing filters, and detecting spam mail [41, 32, 42, 56, 25]. With users of news services and email inundated with information, it is often important to rank or present those pieces of data that are most important to the user. These systems typically make use of previous actions or input by the user to learn user needs and preferences.

- Document organization: There are many applications for which documents that are obtained must be placed into one of many categories for the purpose of easy indexing or access. Examples of this include placing the abstracts of submitted papers to a conference under a correct keyword or subtopic, or placing classified advertisements under the correct heading in a newspaper [64, 51, 17]. This would also include categorizing Web pages into hierarchies. It is difficult, if not impossible, for the large number of Web pages that exist to be manually classified into the proper area of a search engine hierarchy. Automatic text categorization methods are typically used for this purpose[27].

Some of the first approaches to automatic text classification were rule-based [29], where a set of rules was manually developed and applied to documents to determine document classes. For example, an abstract from a technical paper in a physics journal might be determined to be about the subfield astro-physics if it contains the words *planet* or *galaxy* or *stars*. These rules must be formed for each domain where classification is needed. This method is extremely labor intensive and not easily amenable to changes and updates. Instead of hand-crafting rules, the machine learning community approaches text-categorization problems as "supervised" learning problems. In this case the human expert simply has to label a set of examples with appropriate classes. Once a corpus of correctly labeled documents is available, there are a variety of techniques that can be used to create a set of rules or a model of the data that will allow future documents to be classified correctly. The techniques can be optimized and studied independently of the domains and specific problems that they will be used to address.

The problem with the supervised learning approach to text classification is that often very many labeled examples (or "training examples") must be used in order for the system to correctly classify new documents. These training examples must be hand-labeled, which might be quite a tedious and expensive process.

1.2 Beyond Supervision

Suppose we have a text classification problem with an insufficient number of training examples to create a classifier that has a reasonably low error rate on unseen test examples. There are a number of approaches that may be taken to aid in the creation of a more correct classifier.

- The simplest approach would be to insist upon being provided with many more labeled training examples before creating an accurate classifier. This is often an unrealistic possibility. A large numbers of examples may be unavailable, or it may be merely a too expensive and labor intensive project to obtain the number of training examples that are necessary. An approach that has been taken by a number of researchers has been to choose, in some smart way, a *small* number of additional training examples that should be hand-labeled in the hope that this selection will improve learning. Uncertainty sampling has been used in this way [40] where specific examples are chosen out of a large pool of unlabeled examples to be given to humans to be classified. These hand labeled examples then become part of the training corpus. These examples are chosen based upon the inability of the current classifier to label them with high confidence. In this way fewer examples must be given to an expert to be labeled than if the examples were simply randomly sampled.

- Many researchers have noted that although it is often the case that there are very few labeled examples, there are often many unlabeled examples readily available [2, 40, 6, 51, 5, 28]. Even if we do not wish to give these unlabeled examples to experts to label, they can be used in various ways and in conjunction with a variety of classifiers to improve classification. Current work using naive Bayes text classifiers use the labeled training examples to assign probabilistic classes to many unlabeled examples. These newly classified examples are then used in the classifier creation process [51, 66]. Unlabeled examples have also been used to create new features for the set of labeled examples. In this way the set of labeled examples is enhanced by the information provided in the set of unlabeled examples [58]. When a small set of training examples can be re-expressed using different views [6, 50, 15], or if two distinct learning algorithms can be used on the same small set of training data [28], the combination of the labeled and unlabeled sets can be used to achieve a highly accurate classifier. Empirically it has been shown that combining labeled and unlabeled examples can improve accuracy on unseen test sets quite dramatically in certain situations [6, 48, 51, 31, 65, 7, 66].

- In general, the model of text classification that we have described only assumes the existence of a corpus of labeled examples. Once the model is created it is then applied to the test examples. However, if the test examples are available during the model creation process, they can sometimes be exploited to improve learning. The test examples themselves can be looked at as unlabeled examples, and often they are even more useful than a random corpus of unlabeled examples. Since we wish to achieve high accuracy on exactly this test corpus, it has been realized that it can be used within the classifier creation process. Transduction, as defined by Vapnik [61], as opposed to induction, is the creation of a classifier that takes into account the test set to which it will be applied. Transduction has been applied to the text categorization problem using support vector machines [33], where the test examples are used to choose between classifiers that perform comparably on the training data alone.

- Our approach to solving the problem of limited training data is different in some important aspects from those three described above. We do not assume the existence of either unlabeled examples or test examples. Rather we focus on using some related corpus of data in addition to the labeled training set. The question that we address is as follows: Given a text categorization task, can we possibly find some *other* data that can be incorporated into the learning process that will improve accuracy on test examples while limiting the number of labeled training examples needed? We believe that the answer is most often "yes". For example, suppose that we wish to classify the names of companies by the industry that it is part of. A company such as *Watson Pharmaceuticals Inc* would be classified with the label *drug*, and the company name *Walmart* would be classified as type *retail*. Although we many not have numerous training examples, and the training examples are very short, we can find other data that is related to this task. Such data could be articles from the business section of an on-line newspaper or information from company home pages. As a result of the explosion of the amount of digital data that is available, it is often the case that text, databases, or other sources of knowledge that are related to a text classification problem are easily accessible. We term this readily available information "background knowledge". Some of this background knowledge can be used in a supervised learning situation to improve accuracy rates, while keeping the hand-labeled number of training examples needed to a minimum.

1.3 What is Background Knowledge?

Suppose that you were maintaining a Web site about veterinary issues and resources. One of your design decisions has been to divide Web pages according to the animals that they are about. You might have a site devoted to primates, and one for cows, one for dogs, etc., each of which has a list of Web page titles that are associated with that particular animal. In this way, a person browsing your site would have access to many different pages on his/her topic of interest. A text classification problem related to this task might be placing Web page titles in the appropriate list. For example, the Web page entitled "Mad Cow Panic Goes Beyond Europe" (http://archive.nandotimes.com/newsroom/nt/morecow.html), would clearly fit under the category "cow".

Let us formulate this classification task as a supervised learning problem. Given a list of Web page titles that have been hand classified (these are the training examples), we wish to create a classifier that will automatically classify new titles. A schematic view of the this problem with a few training examples (taken from http://netvet.wustl.edu) can be seen in Figure 1.1. If there are not many training examples, this problem becomes one that is very difficult. This is because the training examples are quite short, and do not contain very many informative words. It is often the case that a new test example contains words that do not occur in the training examples at all. An automatic classifier based solely on the given set of training examples can not use these new words in the classification decisions.

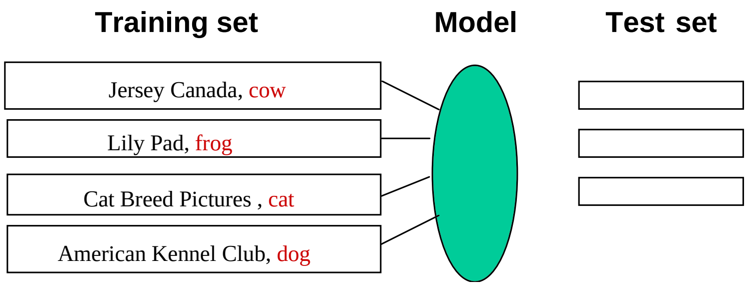
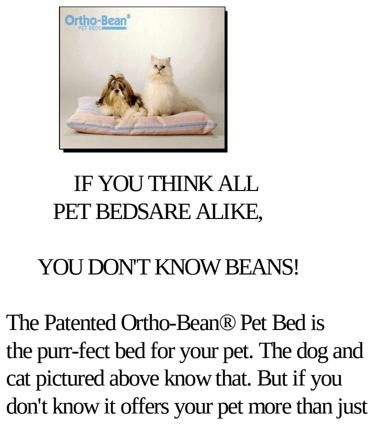


Figure 1.1: The NetVet classification problem

However, we can assume that the World Wide Web contains much information about all the topics and classes that we are dealing with. We can use some of this information as background knowledge for the task. An example of a piece of background knowledge for this task is in Figure 1.2. This Web page, which advertises a pillow for pets, is clearly related to our text classification problem. However, it is important to note that it does not fit clearly into any one of our predefined categories. What then can a piece of background knowledge such as this one add to to the text classification task? Background knowledge can give us information about the co-occurrences of words, as well as the frequency of different words in the domain. For example, from the Web page discussed above, we might learn that the word "pet" is often used in conjunction with "cat" or "dog", but not with "primate" or "cow". The background knowledge can also enhance the sometimes meager vocabulary of the domain that has been created by using only the training examples. Especially when there are very few training examples, the size of the vocabulary that is created from the training examples is small as well, and the chance that test examples will contain words that have not been seen by the learning algorithm, and are hence not part of the model, is very high. In this case the background knowledge enriches the domain model. Depending upon the supervised learning algorithm that is used to classify new test instances, the information gleaned from the background knowledge may be potentially used to improve classification accuracy.


Ortho-Bean®
PET BEDS

IF YOU THINK ALL
PET BEDSARE ALIKE,

YOU DON'T KNOW BEANS!

The Patented Ortho-Bean® Pet Bed is
the purr-fect bed for your pet. The dog and
cat pictured above know that. But if you
don't know it offers your pet more than just

Figure 1.2: An example of background knowledge for the NetVet classification problem

1.4 Contributions

In this thesis we explore three different methods of automatic text classification that have been widely used and have been shown to be competitive techniques in the machine learning and information retrieval literature. For each of these procedures we provide a framework for the inclusion of background knowledge. The methods makes different assumptions about the background knowledge and use the added knowledge in qualitatively different ways.

For the first method, we simply borrow the approach that others have taken [51] in using unlabeled examples to improve text classification. We then broaden their results by applying their method to exploit background knowledge. We create a naive Bayes classifier based upon the training examples alone. This classifier is then used to classify the background knowledge by providing the probability that each piece of background knowledge belongs to each class. These pieces of background knowledge are then treated as training examples and are added into the original set to create a new classifier. This method had been used with unlabeled examples, with the implicit assumption that each unlabeled example definitely fits into one of the classes of the problem. Our observation is that background knowledge from the same domain can be helpful in this framework as well. This is the case even when the pieces of background knowledge do not clearly correspond to any one of the classes that are used to label the training and test sets.

The next two systems present new and different uses of background knowledge. For the second method we incorporate background knowledge into a nearest neighbor classification algorithm. Without background knowledge, the algorithm finds those training examples that are closest in proximity (using information retrieval metrics) to a new test example. These close neighbors then vote to determine the class of the test example. With the addition of background knowledge, we once again find those training examples that are closest to the test example. However, we do so by finding the pieces of background knowledge that are closest to the test example, and comparing those background pieces to the training corpus. In this way, a training and test example are not compared directly but are considered close if there exists a background piece of knowledge that is close to both the training and test example. Many such close training examples are found, and they vote for the final classification result. To accomplish this indirect comparison we use the text database tool WHIRL [11, 12], and formulate our comparisons as queries in its SQL-type language.

Lastly, we use background knowledge with Latent Semantic Indexing [18, 21] to enhance the corpus of training data and produce a reduced space that models a richer set of semantic dependencies. Once again a nearest neighbor paradigm is used. When background knowledge is not available, singular value decomposition is used to re-express the space that is represented by the training examples. Test examples are expressed in this same reduced space and are then compared to the training examples. Those training examples that are closest to the test example in this new space vote to obtain the final results of classification. We incorporate background knowledge by using it in the singular value decomposition process. We add the background knowledge to the training data before the space reduction is performed. This results in the re-expression of both training and test data in a space that was created using the information contained in the background knowledge. They can then be compared as if background knowledge was not used. Since the purpose of Latent Semantic Indexing is to find relationships between words in the corpus, the inclusion of background knowledge allows us to model relationships that cannot be found in the training set alone.

1.5 Outline

The remainder of this thesis is organized in the following manner:

Chapter 2 presents our experimental methodology, and introduces the data sets that we use for evaluation. We discuss the sources of the data and the background knowledge and provide examples and descriptions of each of them.

The next three chapters present the supervised learning methods that we use, and describe how background knowledge is incorporated into the learning process. Chapter 3 reviews the work on naive Bayes and expectation maximization and shows that it can be used with background knowledge as well as with unlabeled examples. We provide a discussion of some of the related work in this field.

Chapter 4 presents our work on incorporating background knowledge into a nearest-neighbor text classification system that is built off the data-base tool WHIRL [11, 12, 14]. We give a detailed description of WHIRL and discuss its adaptation to text classification with and without the use of background knowledge.

Chapter 5 describes Latent Semantic Indexing [18], our method of using it for text classification, and the inclusion of background knowledge into the learner. Once again, we use the same data-sets to compare learning with and without background knowledge.

We provide a comparison of the three different systems and an analysis of where and when each performs most credibly in Chapter 6. Chapter 7 includes a discussion of related work. We then conclude with a summary of our contributions and directions for future work in Chapter 8.

Chapter 2

Data Sets and Methodology

2.1 Data Sets

We have tested the systems that we present with numerous different text-categorization problems, which vary in many of their characteristics, as well as in the nature of the background knowledge that is used to aid the task. In this section we give a description of all of the data sets that we have used, including the source of the data and background knowledge, size of the data sets, average length of examples, and previous uses of the data sets in the machine learning community.

2.1.1 20 Newsgroups

Two of the data sets that we have used have been widely studied by other researchers in machine learning, the 20 Newsgroups data and the WebKb dataset (http://www.cs.cmu.edu/˜textlearning).

The 20 Newsgroup data set consists of articles from 1993 that were collected by Ken Lang from 20 distinct newsgroups on UseNet [35]. It is hard to distinguish between some of these newsgroups, because some of their topics are very related. There are three newsgroups that discuss politics (talk.politics.mideast, talk.politics.misc and talk.politics.guns), five newsgroups that are computer related (comp.os.ms-windows.misc, comp.sys.ibm.ps.hardware, comp.sys.-mac.hardware, comp.windows.x), four science newsgroups (sci.crypt, sci.electromics, sci.med, scip.space), three about religion (alt.atheism, soc.religion.christian, talk.religion.misc), four on recreational activities (rec.autos, rec.motorcycles, rec.sport.hockey, rec.sport.baseball) and one miscellaneous newsgroup (misc.forsale). The vocabulary consists of 62258 words that are each used more than one time [51, 49]. Although it might sometimes be easy for an automated machine learning system to distinguish between newsgroups that are dissimilar, this problem becomes very hard when trying to determine which of a few related newsgroups an article falls into. This problem is particularly challenging when there are very few labeled examples, as that makes it even harder to distinguish between classes.

The training set and test set consist of articles from the 20 newsgroups, and the set of background knowledge consists of individual articles from the 20 newsgroups as well. This background knowledge is therefore exactly of the same form and from the same source as both the training and test data. In this case we can alternatively use a more common term for the background knowledge, “unlabeled examples”. Although this is a very limited form of background knowledge, and we must make the assumption that unlabeled examples are readily available, this data set is a good test bed for the comparison of our work to that of other researchers. An example can be found in Figure 2.1.

file software software lord fax de output montreal
montreal quebec time postscript
voice box canada station kind rs pd produce produce
future travel hardcopy ecole
hpgl polytechnique gaetan lasetjet

Figure 2.1: Example from 20 newsgroups

For this data set, we followed the exact train/test/unlabeled example split that Nigam et al. [51, 49] have used. The UseNet headers of all examples that have the name of the correct newsgroup posting are of course removed before any learning is done. A list of stop-words was also removed from all the examples. The latest 200 examples in each group were taken for the test set, to create a test set of 4000 examples. This would closely mimic a natural scenario of training on older articles and testing on current ones. An unlabeled set of 10000 examples was created by randomly choosing these from the 80% remaining examples. Training sets of sizes 20 (1 per class), 40 (2 per class), 60, 100, 200 and 400 were created by randomly choosing from the remaining examples. Ten sets of non-overlapping training examples of the same size were formed and the results that we report are the averages of all of these runs. The results that we present on this data set are done on exactly the same splits as in [51, 49].

2.1.2 WebKb

The WebKb data set consists of pages that are part of computer science departments from many different universities and are placed in one of seven categories: student, faculty, staff, course, project, department, and other. Following the methods of other researchers our data consists of only the 4199 pages that are in the four most common (excluding other) categories of faculty, course, student, and project [51, 33]; our text-categorization problem is to place a web page into one of t

so that n

be seen

office rainbowthreedigit rainbowthreedigit
rainbowthreedigit hours rainbowonedigit
rainbowonedigit rainbowonedigit note this for
and of information homework homework
homework is is is class lecture notes exams
grading description rainbowfourdigit fall
assignment welcome general syllabus
assignments project project announcements it
solutions know systems systems me current

Figure 2.2: A piece of data from WebKb

For purposes of training and testing the data is divided by the department that the web page is associated with. There are five total sets: pages from Cornell, Washington, Texas, Wisconsin and a miscellaneous group that includes many other universities. In this way, four test sets are formed by taking pages from each of the four universities. The test set for Cornell consists of 226 pages, Wisconsin has 308, Washington has 255 and Texas has 251. For each test set, the training and unlabeled sets are taken from the pages of all the other universities. For example, the training and unlabeled examples used with the Wisconsin test set come from Cornell, Washington, Texas and the miscellaneous group. This helps avoid learning parameters that are specific to a university and testing on that same university, as opposed to learning parameters that are specific to the classes.

For each of these four test sets, a set of 2500 unlabeled examples is randomly chosen from all documents of every other university. Once again, for each of the four test sets, from the remaining documents, training sets of size 4 (1 per class) 8 (2 per class), 12, 20, 40, 80, 120, 160, 200 are formed. The results that are reported in this thesis are averages of the 40 runs (10 per data set size per 4 departments). Preprocessing of the data follows other previous methodology, numbers and phone numbers are converted into tokens, and feature selection is done by keeping only the 300 words that have highest mutual information with the classes.

For this data set as well, our split of the articles into the train, test and unlabeled sets is exactly that of [51, 49].

2.1.3 Advertisements

Following the methodology of the 20 Newsgroups and WebKb data sets, we created a data set of short classified advertisements off the World Wide Web. In this set, and other descriptions that follow, the background knowledge no longer consists simply of unlabeled examples. For the labeled set of examples, we downloaded the classified advertisements from one day in January 2001 from the Courier Post at http://www.southjerseyclassifieds.com. The Courier Post online advertisements are divided into 9 main categories: Employment, Real Estate for Sale, Real Estate for Rent, Dial-A-Pro, Announcements, Transportation, Pets, Employment, and Business Opportunity. As in the 20-Newsgroups and WebKb dataset, we created 10 random training sets with 1 per class, 2 per class, 5, 10, 20, 50, and 100 per class. The Courier Post online did not contain the same number of advertisements in each of these classes. For the larger data sets there were not necessarily the same number of examples in each classes as there were not enough advertisements in some of the classes to fit the quota. For example, the class “Announcements” had only 4 advertisements in total, so that class was less represented in the data sets with 5, 10, 20, 50 and 100 examples per class. This mimics the natural world a bit better than having all classes artificially made to contain the same number of examples, so it did not concern us. For testing we simply downloaded advertisements from the same paper, from one day a month later, taking approximately 1000 (25%) of the examples for our test set. In this set the most populous class was *Employment* with 453 examples, *Transportation* had 171, *Real Estate for Rent* 105, *Real Estate for Sale* 61, *Dial-A-Pro* 41, *Pet* 38, and the class *Announcements* had no examples.

The background knowledge from the problem came from another online newspaper – The Daily Record (http://classifieds.dailyrecord.com). The Daily Record advertisements online are divided into 8 categories: Announcements, Business and Service, Employment, Financial, Instructions, Merchandise, Real Estate and Transportation. We treated the union of the articles from each one of these categories as a separate piece of background knowledge. The average length of the training and test data is 31 words; the background information had an average of 2147 words each. The vocabulary size of the full set of data without the background was 5367; with the background it became 6489. A sample example can be seen in Figure 2.3. A portion of one piece of background knowledge (preprocessing has been done to stem and change phone numbers) can be seen in Figure 2.4. Here, the background knowledge is a sequence of numerous different single ads, all about *transportation*.

```
ACCOUNTANT/ Tax Preparer, F/T, exp in Quick books helpful.
Computer exp a must.
Exc. comp with oppty for quartly & yr end bonuses
856-232-0958 or fax: 856-232-7421
```

Figure 2.3: An example from the advertisement data

```
2 cadillac 93 eldorado tour coup
north star system polo green w beig
lthr int bose stereo cd ex cond
ask 7900 phonenum
cadillac 90 sedan devill 1 owner 163k
run well  1295   phonenum
chevi 95 cavali auto ac 4 cyl p pb
tilt rr def am fm   stereo cd 88 736
mi stk l3823A vin s7145937  499
0 lic tax  mv  fee ex tra saturn of
denvill phonenum
chevi 93 corsica v6 4dr auto 79k load
1 owner like new 2975  phonenum
```

Figure 2.4: Part of a piece of background knowledge from the advertisement data

2.1.4 ASRS

The Aviation Safety Reporting System (http://asrs.arc.nasa.gov/) is a combined effort of the Federal Aviation Administration (FAA) and the National Aeronautics and Space Administration (NASA). The purpose of the system is to provide a forum by which airline workers can anonymously report any problems that arise pertaining to flights and aircraft. The incident reports are read by analysts and classified and diagnosed by them. The analysts identify any emergencies, detect situations that compromise safety, and provide actions to be taken. These reports are then placed into a database for further research on safety and other issues. We obtained the data from http://nasdac.faa.gov/asp/ and our database contains the incident reports from January 1990 through March 1999.

Since we are interested in text categorization tasks, there are two parts of each incident report that we deal with, the “narrative” and the “synopsis”. The “narrative” is a long description of the incident, ranging in our training and test data set from 1 to 1458 words with a median length of 171 words. The “synopsis” is a much shorter summary of the incident with the length ranging from 2 to 109 with a median length of 24. It is interesting to note that many of these words are sometimes abbreviated which makes the text classification task even harder.

The are many different categorization problems that can be taken from this data set. A feature that is associated with each incident is the consequence of the incident that the analyst adds to the report. This can take on the values: aircraft damaged, emotional trauma, FAA investigatory follow-up, FAA assigned or threatened penalties, flight control/aircraft review, injury, none, and other. If more than one consequence was present we removed that incident from our training and test data. We also removed from the training and test data all those incidents that had categories of *none* and *other*. This then became a six class classification problem.

We chose the training and test sets to consist of the *synopsis* part of each incident. The test set consists of data from the year 1999 for a total of 128 examples. The class *review* had the most number of examples at 56, and the class *FAA assigned or threatened penalties* had the fewest at only 2 examples. The training set consists of all data from 1997 and 1998, for a total of 591 examples. For the background knowledge, we chose all *narratives* from 1990-1996. In this case we did not remove any examples; thus the background knowledge contains those reports whose categories were *other* and *none* as well as the six that are found in our training and test set. For this data set, therefore, the training and test examples are shorter than the background pieces of knowledge, and the background pieces do not all fit into the categories of the text classification problem. The total vocabulary size of the training and test data combined was 1872; when the background knowledge was added the vocabulary had a total of 3771 words. An example of a training or test instance can be seen in Figure 2.5, and a piece of background knowledge can be seen in Figure 2.6. Once again, as in the 20-Newsgroups and WebKb dataset, we created from our training data 10 random training sets with 1 per class, 10 sets with 2 per class, and 10 sets with 5, 10, 20, and 50 per class.

```
A DC10 ENTERS AN AREA OF PRECIPITATION
AND TURB WITH ALL PAX AND FLT ATTENDANTS
STRAPPED IN. 5 PAX LATER COMPLAIN OF BACK
INJURIES
```

Figure 2.5: An example from the ASRS data

```
I WAS ON LOW LEVEL FLT FOR THE PURPOSE OF
OBSERVING PORPOISE AND SEALS ALONG THE
 SHORELINE AND THE OCEAN, ACCORDING WITH
THE FAR 91 79 POINT A AND C
FLYING CROSS-COUNTRY IN MVFR  ABOUT 10 MI
SW OF DEST, CEILING STARTED TO DROP AN
D SO DID VISIBILITY  NAVIGATION WAS BEING
ACCOMPLISHED BY FOLLOWING MAJOR ROADS
 DUE TO VISIBILITY LIMITATIONS, I MISSED AN INTXN
TURN  AFTER DISCOVERING PROBLE
M, I STOPPED CRUISE AND HOVERED TO READ
 LARGE INTERSTATE OVER THE ROAD SIGN
GEAR UP LNDG  CAUSE: PLT S MOMENTARY DISTR
 IN LNDG PHASE OF FLT I WAS INSTRUCTING IN A VFR
ACFT AT AN UNCONTROLLED ARPT AND CALLED THE
ACFT WHILE JUST TOOK OFF TO ASK FOR CEILING  ACFT
 RPTED 1000   I TOOK OFF AND DETERMINED
THE CEILING WAS 1000  MSL, A TRUE CEILING OF 300  AG
 I HAD TO DECLARE AN EMER BECAUSE OF LOW
CEILING AND RAPIDLY DECLINING VISIBILITY
```

Figure 2.6: A piece of background knowledge from the ASRS data

2.1.5 Physics Papers

One common text categorization task is assigning topic labels to technical papers. We created a data set from the physics papers archive (http://xxx.lanl.gov), where we downloaded the titles for all technical papers in the first three areas in physics (astrophysics, condensed matter, and general relativity and quantum cosmology) for the month of March 1999. As background knowledge we downloaded the *abstracts* of all papers in these same areas from the two previous months – January and February 1999. In total there were 1701 pieces of knowledge in the background set, and 1066 in the training-test set combined. The distribution of classes was skewed, however, as there were 493 titles in astrophysics, 460 in condensed matter, and only 113 in quantum cosmology. These background knowledge abstracts were downloaded without their labels (i.e., without knowledge of what sub-discipline they were from) so that our learning programs had no access to them. We created two problems from this set; a two-class problem where we included only those titles and abstracts from the first two areas (astrophysics and condensed matter). These two classes had almost the same number of training and test examples. The three-class problem included the titles and abstracts from the third category (quantum cosmology) as well.

The average length of the technical paper titles was 12.4 words while the average length of the abstracts was 141 words. The number of words in the vocabulary taken from the full set of titles was 1716; with the abstracts it went up to 6950. Two examples of titles from the training/test set are in Figure 2.7, and an abstract from the background knowledge can be see in Figure 2.8.

Stellar populations in the Phoenix dwarf galaxy
M4-18: The planetary nebula and its WC10 central star

Figure 2.7: Examples from the physics domain

We consider high-temperature expansions for the free energy of zero-field Ising models on planar quasiperiodic graphs. For the Penrose and the octagonal Ammann-Beenker tiling, we compute the expansion coefficients up to 18th order. As a by-product, we obtain exact vertex-averaged numbers of self-avoiding polygons on these quasiperiodic graphs. In addition, we analyze periodic approximants by computing the partition function via the Kac-Ward determinant. For the critical properties, we find complete agreement with the commonly accepted conjecture that the models under consideration belong to the same universality class as those on periodic two-dimensional lattices

Figure 2.8: Background example from the physics domain

2.1.6 NetVet

Two of the data sets that we have used are taken from the work of Cohen and Hirsh [14]. They created these data sets from sites on the World Wide Web, to test the WHIRL database system as a nearest neighbor classification system. The first data set, NetVet (http://www.netvet.wustle.edu) included the Web page headings for the pages concerning cows, horses, cats, dogs, rodents, birds and primates. The text categorization task was to place a web page title into the appropriate class. For example, a training example in the class birds might have been: “Wild Bird Center of

Walnut Creek". Each of these titles had a URL that linked the title to its associated Web page. For the training/test corpus, we randomly chose half of these titles with their labels, in total 1789 examples. Of these examples, 280 were of class horse, 289 of class bird, 518 of class dog, 241 of class cat, 136 of class rodent, 252 of class cow, and 73 of class primate. We discarded the other half of the titles, with their labels, and simply kept the URL to the associated Web page. We used these URLs to download the first 100 words from each of these pages, to be placed into a corpus for background knowledge. Those URLs that were not reachable were ignored by the program that created the background knowledge. In total there were 1158 entries in the background knowledge database.

In this case, the background data consisted of much longer entries than the training and test data. The titles had an average length of 4.9 words, while each piece of background data contain 100 words. The total vocabulary size of the titles was 1817 words; it jumped to 10399 words when the background data was added. However, the words in the background data were not as informative as the shorter title strings, because people tend to place important words in their titles. Many of the background pieces had words unrelated to the task at all. For example, a number of them began with the words: "Welcome to the site", which is not specific to any category, or the domain at all. Training and test examples can be seen in Figure 2.9 and Figure 2.10 shows an example of a piece of background knowledge.

Cornell Univeristy Equine Research
Disease of Primates

Figure 2.9: Training and test examples from the NetVet domain

Friends don't let friends run in wire wheels.
Transoniq offers three exercise wheels, Wodent Wheels(TM), for small critters. Wodent Wheels feature a unique design with a safe more natural running surface and a support stand without dangerous pinch areas. Available in three sizes: 8-inch diameter, 11-inch diameter, and 12-inch diameter. The only excersise wheel recommended and approved by the ASPCA. Rodents are attracted to the wheels entry holes which offers them a real nifty effect when watching them run. Even rats love wheels if they don't get their feet hurt and their tails whacked by unsafe axle support.

Figure 2.10: Background example from the NetVet domain

2.1.7 Business Names

The second of Cohen and Hirsh's data sets consisted of a training set of company names, 2472 in all, taken from the Hoover Web site (http://www.hoovers.com) labeled with one of 124 industry names. The class *retail* had the most business names associated with it; at 303 examples, and there were a few classes with only one example each (*coal*). The average number of examples per class is 20. We created background knowledge from an entirely different Web site, http://biz.yahoo.com. We downloaded the Web pages under each business category in the Yahoo! business hierarchy to create 101 pieces of background knowledge. The Yahoo! hierarchy had a different number of classes and different way of dividing the companies, but this was irrelevant to our purposes since we treated it solely as a source of unlabeled background text. Each piece of background knowledge consisted of the combination of Web pages that were stored under a sub-topic in the Yahoo! hierarchy. Each example in the training and test set had an average of 4 words (for example: Midland Financial Group, Inc). The instances in the table of background knowledge had an average length of 6727 words and each one was thus a *much* longer text string than the training or test examples. Vocabulary size for the business names was 2612 words; with the background knowledge it was 22,963 words.

A few training/test examples from this domain are in Figure 2.11. A small part of one of the very large pieces of background knowledge is in Figure 2.12. Each single piece of background knowledge consists of a discussion of *many* businesses such as the one in Figure 2.12.

Yale Univeristy
Xerox Corporation
Abbot Laboratories

Figure 2.11: Training/test examples from the business name data

2.1.8 Clarinet News

Another data set that we created was obtained from Clarinet news. We downloaded all articles under the sports and banking headings on November 17, 1999, using the most recent ones for training and test sets and the older ones for background knowledge. In total, our background knowledge consisted of a corpus of 1165 articles. The background knowledge in this problem

CET Environmental Service (AMEX : ENV) ENV provides
a variety of consulting and technical services to
resolve environmental and health risk problems
in the air, water and soil with an expertise in environmental
remediation and water treatment techniques. For the three months
ended 3/99, revenues rose 13 percent to $12.9 million. Net income
applicable to Common totalled $346 thousand vs. a loss of $666
thousand. Revenues reflect higher revenue provided EPA by
contracts. Earnings reflect a continued focus on cost control.

Figure 2.12: A part of a piece of background knowledge from the business name data

consisted of the first 100 words of each of these articles. Informal studies showed us that including the entire articles did not improve accuracy substantially, probably because the most informative part of an article is usually the first few paragraphs. Our training-test data had 1033 data points of which 637 belonged to the sports category, and 406 belonged to banking category. We took the first nine words of each news article to create the 1033 examples in the training and test set. The results that we present are on these short text strings. The size of the vocabulary of the training and test set is 1984, but when the background knowledge is included there are 7381 words. Two short text strings from the training data can be seen in Figure 2.13; a piece of background knowledge is in Figure 2.14.

The Baseball Writer's Association of America postseason award Monda
From national championship game to consolation game in just

Figure 2.13: Training/test examples from the Clarinet data

The New York Knicks hope to recover from their fourth
quarter fold in Minnesota on Thursday night when they
visit the Fleetcenter tonight to battle the Boston Celtics.
Allan Houston scored points but the Knicks blew
a point fourth quarter lead and Latrell Sprewell
missed a potential tie pointer at the buzzer in a loss
to the Timberwolves. New York led throughout and
opened a cushion with to play before Kevin Garnett
took over to score for a career high point for Minnesota.

Figure 2.14: A piece of background knowledge from the Clarinet data

2.1.9 Thesaurus

Roget's thesaurus places all words in the English language into one of six major categories: space, matter, abstract relations, intellect, volition, and affection. For example, a word such as "superiority" falls into the category *abstract relations*, while the word "love" is placed into the category *affection*. Following the six links, one for each of these categories, from http://www.-thesaurus.com, we created a labeled set of 1000 words, with each word associated with one category. The smallest category contained 135 labeled examples and the largest class contained 222 examples. The training and test sets had a combined vocabulary size of 1063 words (as most examples had only one word each), but when the background knowledge was added the total vocabulary size became 11607 words.

We obtained our background knowledge via http://www.thesaurus.com as well, by downloading the dictionary definitions of all 1000 words in the labeled set. We cleaned up the dictionary definitions by removing the sources that were returned (i.e. which dictionary the information was gleaned from) as well as other miscellaneous information (such as how many definitions were found). Each of these dictionary definitions became an entry in our background knowledge database. An interesting point about this data set is that the background knowledge contains information directly about the test set, i.e. definitions of the words in the test set. Since these definitions are not directly related to the classification task at hand, this poses no contradiction. As a matter of fact, we can look at new test examples given to the system as follows: Given a word and its definition, place the definition into the background knowledge data base, and then categorize the word using the total background knowledge. Some words from the training and test set are in Figure 2.15, and a definition from the background knowledge is in Figure 2.16.

Existence
Imitation

Figure 2.15: Training/test examples from the thesaurus data

2.2 Testing Methodology

For the first two domains that we described in Sections 2.1.1–2.1.2 our testing methodology followed that of Nigam et al. [51]. We obtained our train/test/unlabeled splits directly from their

```
 dear  dr    adj dearer dearest                 a Loved
and cherished my dearest friend          b Greatly valued
precious lost everything dear to them       Highly
esteemed or regarded Used in direct
 address especially in        salutations Dear
 Lee Dawson  a Highpriced expensive
b Charging high prices       Earnest ardent
This good man was a dear lover and constant
practicer of angling  Izaak Walton
Obsolete Noble worthy       Heartfelt
 It is my dearest wish            n       One that is
 greatly loved       An endearing lovable or kind person
  adv       With  fondness affectionately
At a high cost sold their wares dear
 interj           Used as a polite exclamation
chiefly of surprise or di
 stress           oh dear dear me  Middle English dere from
Old English dore           dearly adv    dearness n
```

Figure 2.16: Background knowledge from the thesaurus data

work, and the accuracy results that we graph for the first method (presented in Chapter 3) for these two data sets the same as those in [51]. For the data sets that are described in Section 2.1.3 and Section 2.1.4 we use the same method of obtaining the test and train sets. The test set is taken randomly from later examples, and different size training sets are created by randomly choosing an equal number of examples from each class. Ten different training sets are created for each size; the results of these ten sets are averaged together to arrive at a final results.

Our testing methodology for the other data sets that we used to test our systems was slightly different [64, 65]. In each of the cases, we either obtained from other researchers or created from the Web a labeled set (from which we created both the training and the test set) and a set of background knowledge. All reported results are five-fold cross-validated. The labeled set is divided into a training set and a test set by choosing a random 80% of the labeled data for the training set, and the remainder of the labeled data for the test set. The code that we used to create these sets came from the shell script associated with the C4.5 program by R. Quinlan [53]. This set creation is done five times to obtain five non-overlapping test sets, and our final results are the averages across these five runs. For each of these five runs, we tested our system using 20%, 40%, 60%, 80% and 100% of the current training set. This gives us a feeling of the usefulness of background knowledge as the size of the training set is varied.

2.3 Summary

We have presented nine different data sets on which we tested our systems. All the problems are text classification problems to be used with background knowledge. However, the background knowledge that we used for these tasks vary greatly. For two of the data sets, 20 newsgroups and WebKb, the background knowledge is simply a set of unlabeled examples. Two of the data sets, business names and advertisements, have background knowledge that is of a very different structure than the training and test sets. For these two data sets as well as for the ASRS data set, the background knowledge pieces do not fit clearly into any of the classes of the text categorization task. The physics, NetVet, and thesaurus data sets have background knowledge that is of a different type than the training and test set, but that can fit into the classes of the text classification task. The Clarinet newsgroup data set contains background knowledge that is from the same source and of the same form as the training and test set, but is just much longer.

Chapter 3

A Generative Model with Background Knowledge

Generative models for text categorization assume that some statistical process has produced the documents that are available for the classification task. Given a set of training documents, each of which has an assigned class, generative models estimate the parameters of the generative process [51, 39]. These parameters are chosen based upon the set of training data so that it is likely that the generator could have created these exact pieces of training data. If, in addition to the classified training data, we have data of the same type that is unlabeled, there are methods that can be used to estimate the parameters of the generative models while incorporating these unlabeled examples [51]. We show in this chapter that these unlabeled examples need not be exactly of the same type and form as the training set. We can substitute more general background knowledge for the unlabeled examples, and using the methods of other researchers, obtain improvements in accuracy on text classifiers that are created using both the training set and the set of background knowledge. In essence, we empirically show that background knowledge can be used to improve accuracy in a generative model.

3.1 Naive Bayes

One such generative model is the naive Bayes method. This learning method has been studied quite carefully in both the information retrieval and machine learning communities, and in practice performs very well on text classification tasks [55, 38, 41, 39, 43, 37, 34, 47, 36, 45, 51]. Given a set of training documents, $\{x_1, \ldots, x_n\}$, each of which is assigned one of m classes $\{c_1, \ldots, c_m\}$, the probability of any word occuring given class c_j can be estimated from the training data. This is typically the number of times that the word occurs in this class divided by the total number of words in this class. Laplace smoothing is often used to prevent zero probablilities of unseen events where one is added to the numerator and the number of terms is added to the denominator. The training examples can therefore be used to compute the probability that a given document (or a sequence of words), x_k, will occur given the class c_j. If the document x_k consist of the words $w_{k,1}, \ldots, w_{k,d}$, then assuming that all words are independent of each other, the equation to compute the probability that this document will occur given the class c_j, can be given as follows:

$$P(x_k|c_j) = \prod_{i=1}^{d} P(w_{k,i}|c_j) \tag{3.1}$$

The probability of a class c_j occuring can also be estimated from the training data. This is typically the number of documents in class c_j divided by the total number of documents. Once again Laplace smoothing can be used, adding one to the numerator and the number of classes to the denominator. The important point to note is that these values are estimated directly from the word counts and class counts of the training data.

Using Bayes rule we can then specify the probability that a specific example, x_k is a member of the class, c_j:

$$P(c_j|x_k) = \frac{P(c_j) \times P(x_k|c_j)}{P(x_k)} \tag{3.2}$$

If we substitute the numerator of Equation 3.2 with Equation 3.1 we have:

$$P(c_j|x_k) = \frac{P(c_j) \times \prod_{i=1}^{d} P(w_{k,i}|c_j)}{P(x_k)} \tag{3.3}$$

During classification time this equation is used on a test document to compute the probabilities of class membership in all classes. We are interested in finding $argmax_j P(c_j|x_k)$, or to use Equation 3.3:

$$argmax_j \frac{P(c_j) \times \prod_{i=1}^{d} P(w_{k,i}|c_j)}{P(x_k)} \tag{3.4}$$

In practice, this reduces to $argmax_j P(c_j) \times \prod_{i=1}^{d} P(w_{k,i}|c_j)$. This the class with the highest probability and is returned as the final classification result. The independence assumption is clearly violated in text because based upon common usage of the English language as well as grammar rules we know that some words are more likely to occur with other words, but this does not seem to harm performance. Although it is often the case that the final probabilities that are returned by the Bayes classifier are not accurate, misclassification error is often minimized.

There has been some theoretical work on the optimality of Bayes classifiers, which has shown that classification accuracy is not highly affected by the independence assumptions [20, 26].

3.2 Naive Bayes and Expectation Maximization

For a model that is created using only labeled data, we have just seen from Equation 3.3 that naive Bayes can be used for classifying text documents. However, when unlabeled examples are available to aid the classification task numerical methods can be used to approximate the values that can no longer be obtained directly from the data.

Dempster has presented an iterative hill-climbing technique, Expectation Maximization, for problems with missing data [19]. The labels of the unlabeled examples can be looked at as missing data [19] and they can be approximated via this algorithm. This algorithm is the one most commonly used for this type of application where data is incomplete [4], and most recently it has been applied to the labeled-unlabeled data problem in text classification [16, 7, 51]. Nigam [49] presents the use of the EM algorithm with a naive Bayes classifier for text classification as follows:

- Compute the initial parameters of the classifier by using only the set of labeled examples.
- E step: Compute the probabilities of class membership for each of the unlabeled documents given the current classifier parameters. This is done by using the current version of the naive Bayes classifier.
- M step: Using the probabilities that were computed in the E step, recompute the parameters of the naive Bayes classifier.

  The E step gives the probability that each unlabeled examples is classified by each class. To reestimate the probability that a class c_k occurs using both the *training* (labeled) set and the *newly labeled examples* (which were the unlabeled set) we no longer calculate the total number of documents in the class divided by the total number of documents. Rather, we calculate the sum of the *probabilities* that all documents belong in c_k divided by the total number of documents. For a document in the training corpus, this probability is equal to one if the document belongs to the class c_k, and zero otherwise. For documents in the newly labeled set this probability is equivalent to the results of the E step. To reestimate the probability that a word will occur given a specific class it is not enough to compute the number of times that the word occurs in each document that belongs to that class, but rather the number of times that the word occurs in each document multiplied by the probability that the document belongs to that class. If an unlabeled example has a non-zero probability of belonging to a specific class, it is used in the calculations for that class. In this way unlabeled examples are actually used numerous times in the recalculation of the model parameters.

The E and M steps are repeated iteratively. Nigam et al. [51] stop the process when there is no change in the model parameters. A tuning set could also be used to decide when to stop the evaluation of parameters of the classifier. Our version of the algorithm iterates for a fixed number of times (seven) that was found to be useful in text classification.[1] For the basis of comparison with the other algorithms that we present in Chapters 4 and 5, we report the highest classification accuracy from among those seven iterations. This skews the results slightly in favor of the EM algorithm as compared to the results that we report in later chapters. In cases where the first few iterations of EM help classification, but there is a subsequent degradation in accuracy as the iterative process continues, we report that as well. We used the rainbow package (http://www.cs.cmu.edu/˜mccallum/bow/rainbow/) [46] to preprocess and tokenize the data and to run naive Bayes and EM.

3.2.1 Unlabeled Examples vs. Background Knowledge

At first glance it would seem that although EM might be a useful technique for aiding the classification task via unlabeled examples, the same technique would be useless when dealing with the much broader problem of using background knowledge. This is because the assumptions that the naive Bayes classifier makes is that examples (both labeled and unlabeled) have been generated by a mixture model that has a one-to-one correspondence with classes. Even if this assumption is true for the labeled data and the test data, by its very nature, background knowledge should not fit this assumption at all. Background knowledge often comes from a source

[1]We chose the number 7 based on discussions with Nigam (personal communication).

that differs from that of the training and test data and is of a different form and different size than the training and test data.

Consider, for instance, the text categorization problem of placing advertisements into the correct area in the classified section of a newspaper. If we have a very large number of previously classified advertisements, this might be a task that is not very difficult for an automated machine learning program. However, if the labeled data is scarce, this becomes a much more difficult problem. For example, a piece of test data might be (taken from http://www.courierpost):

> toyota ’99 tacoma 4x4 x cab load
> must sell 21 000 nego call joe

and belong to the class *truck*. If the set of training data is small, and the term “toyota” is not part of the training set vocabulary, this advertisement might be misclassified. If we have a set of unlabeled examples of advertisements then perhaps naive Bayes and EM could correctly approach the classification problem. However, suppose that our background knowledge consists of sections of advertisements from some other newspaper, where each *section* is a piece of background knowledge. One piece of background knowledge consists of all advertisements under a specific categorization in the second newspaper. Moreover, the grouping in the second newspaper is very different than the first. For example, the second newspaper has one category called *transportation* that combines three categories of the first newspaper – *cars*, *trucks* and *boats*. This piece of background knowledge *should* be helpful, but it clearly violates all assumptions about the generative model, and it does not fit into the classification problem that we wish to learn.

On the other hand, there are many examples where, although the form of the background knowledge is different than the training and test data, the background knowledge may still follow the same classification scheme as the training and test data. Consider the problem of classifying the titles of technical papers in physics by sub-fields. For example, a title (xxx.lanl.gov):

> The Nature of Galaxy Bias and Clustering

would be placed in the category *astro physics*. Suppose, also, that for background knowledge we have numerous abstracts of technical papers available. Although it is the case that these pieces of background knowledge are not short title strings, we can still look at them as possibly falling into one of the categories for classification. Since it is the case that in text categorization all data is represented in the same way, as vectors of terms, in that sense we can still look at the background abstracts as examples with missing class information. Therefore, perhaps naive Bayes and EM would help in a case such as this. The interesting observation that we make is that to gain leverage out of unlabeled examples, the unlabeled data that we have need not be specifically and accurately unlabeled examples. As long as the vocabulary and classification structure closely resembles the training/test data, background knowledge can improve classification accuracy in textual data using the EM algorithm.

There has been some theoretical work on using unlabeled examples with generative classifiers. The provable usefulness of these unlabeled examples is unclear. Castelli and Cover [9] have shown that labeled examples are much more useful than unlabeled examples, and Cozman and Cohen [16] present work that shows that that unlabeled examples can sometimes in fact degrade learning. For generative modeling of classifiers, if the structure of the classifier that is automatically learned is identical to that of the generator of the training, test and unlabeled documents then it has been shown that unlabeled documents will most definitely be helpful [66]. However, this assumption is often unprovable or untrue, even when dealing with unlabeled examples that are extremely “similar” to the labeled data. Certainly with background knowledge that comes from a different source than the training/test data we cannot rely on this theoretical result. Empirically we show in the next section that background knowledge can aid classification.

3.3 Results

Figures 3.1–3.10 present the results of running naive Bayes (with no background knowledge) and EM on the data sets that we described in Chapter 2. Each graph depicts the accuracy (y axis) curve for naive Bayes and for the EM algorithm, as the number of original training examples is varied (x axis). For the graphs in Figures 3.1, 3.2, 3.3 and 3.4 the x axis represents the number of examples in the training set. The x axis of the other graphs gives the total percentage of the training examples that are used (varying from 20% to 100%, as we discussed in Chapter 2).

The graphs on the 20 Newsgroup dataset (Figure 3.1) and the WebKb data set (Figure 3.2) are taken from Nigam et al. [51], and the background knowledge is simply the set of unlabeled examples.

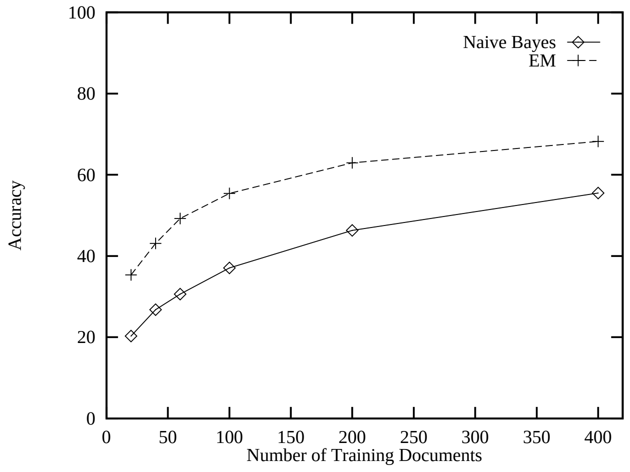


Figure 3.1: Naive Bayes and EM for 20 newsgroups

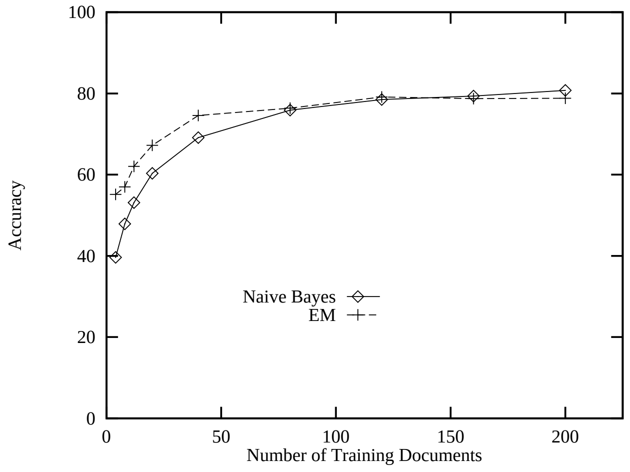


Figure 3.2: Naive Bayes and EM for Webkb

In almost all of the data sets the inclusion of background knowledge helps boost accuracy on the unseen test examples. This improvement is especially noticeable when there are fewer training examples. In general, as the number of training examples increase, background knowledge gives much less leverage. This is consistent with the analysis of other researchers [49, 16], who

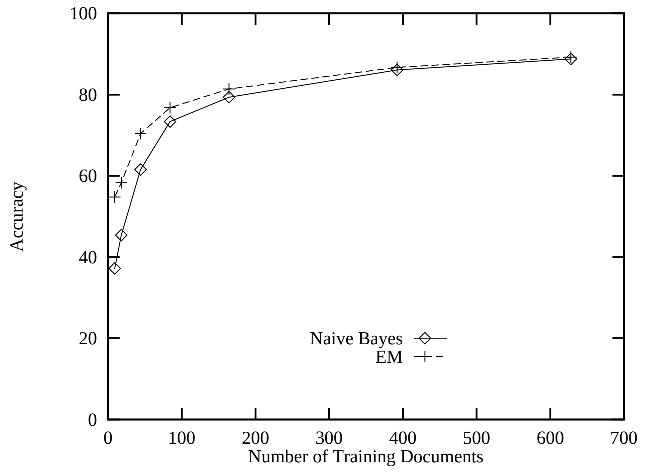


Figure 3.3: Naive Bayes and EM for the advertisements problem

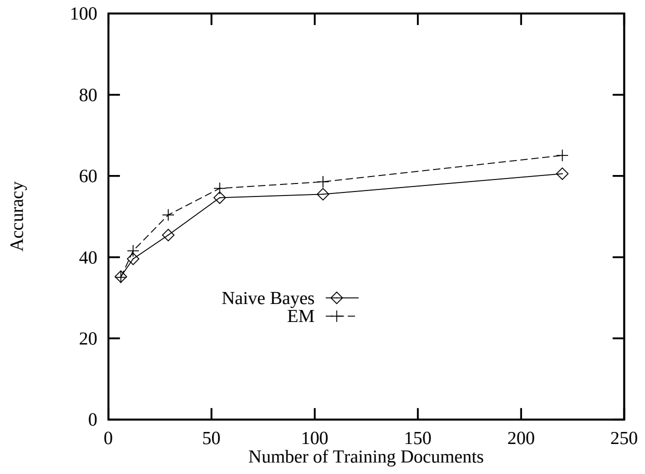


Figure 3.4: Naive Bayes and EM for the ASRS problem

show that additional unlabeled examples are most helpful when the training examples are few. Our interesting observation is that these improvements hold even though the background knowledge is sometimes of a very different form than the training and test example. For the business name data (Figure 3.8) and the advertisement data (Figure 3.3) the classes of the background knowledge are known to be different than the training/test data, yet classification accuracy still improves. The generative model that EM finds need not model the domain properly, as long as the probabilities that it finds is correlated with accuracy.

Is there a significant difference between the accuracy obtained with and without background knowledge? Each x value that is plotted in Figures 3.1–3.10 represents a different data set or size of data set on which naive Bayes and EM was run. To see if EM with background knowledge obtains higher accuracy than naive Bayes, we ran a paired t-test, treating each data set as a seperate trial, with an accuracy associated with it for naive Bayes, and one for EM. The paired t-test deals with the difference between the numbers of each pair of data and the p value gives the probablility that the mean difference is consistent with zero. In this case the resulting p value was less than .01 so we were able to conclude that there is a significant difference in accuracies with and without background knowledge.

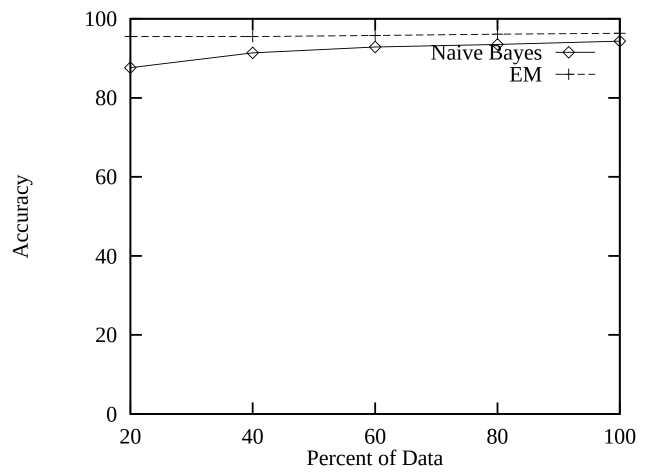


Figure 3.5: Naive Bayes and EM for the 2-class physics title problem

It has been shown that although EM with unlabeled examples can sometimes help accuracy, it can sometimes hurt it as well [49, 16]. Our point to note is not that EM *always* helps, but

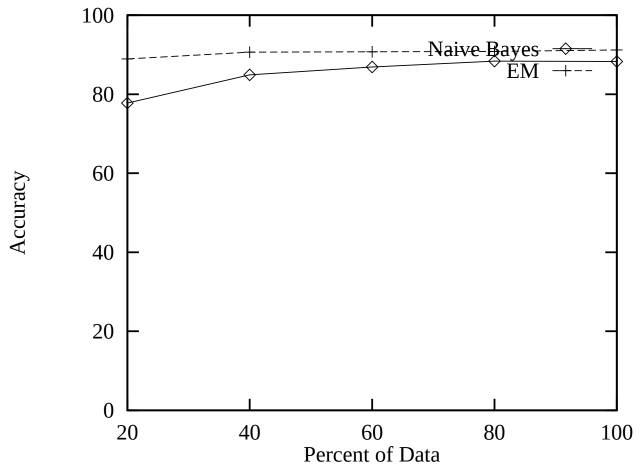


Figure 3.6: Naive Bayes and EM for the 3-class physics title problem

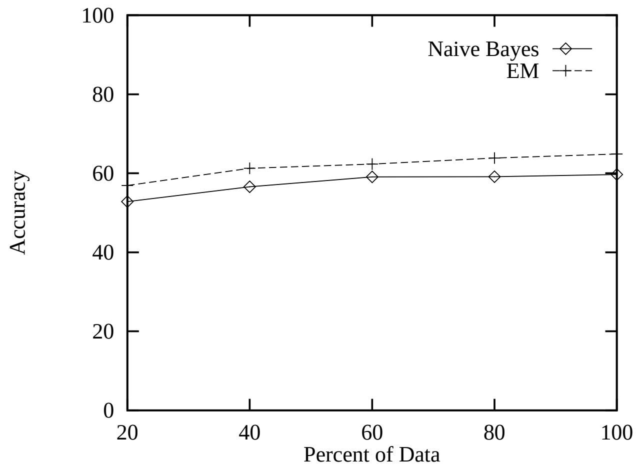


Figure 3.7: Naive Bayes and EM for the NetVet data

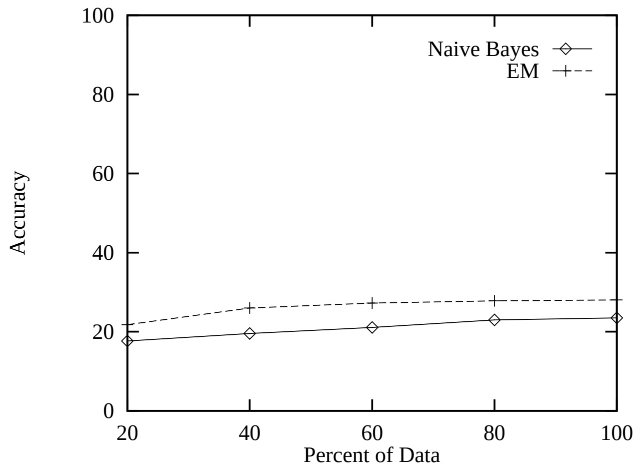


Figure 3.8: Naive Bayes and EM for the business name data

rather that it can help even when broad background knowledge is used instead of unlabeled examples. In particular, the physics paper title problem in Figures 3.5–3.6 and the Clarinet newsgroup problem in Figure 3.9 are really helped by the addition of the background knowledge. We expected this because the background knowledge follows the exact form and classes of the training and test data. However, it was a greater surprise when the thesaurus data set in Figure 3.10 performed quite credibly as well.

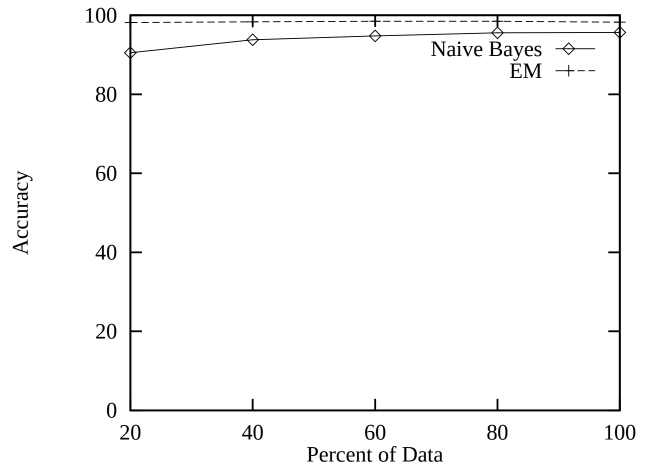


Figure 3.9: Naive Bayes and EM for the Clarinet 9 words problem

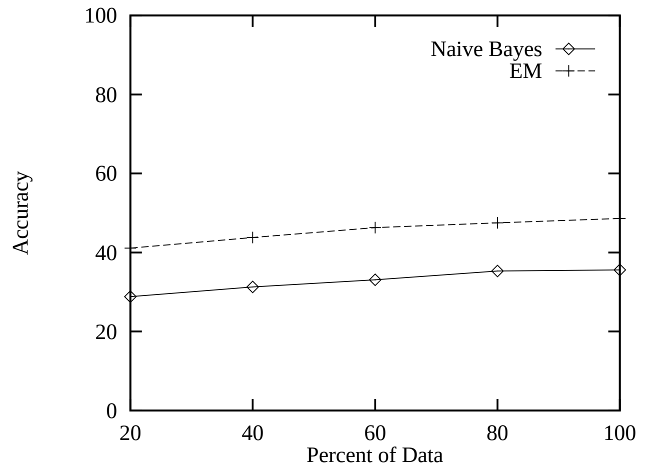


Figure 3.10: Naive Bayes and EM for the thesaurus problem

We mentioned in the previous section of this chapter that the accuracy rates that we have graphed for EM in this chapter correspond to the best value out of the seven EM iterations. In two of the data sets, this value consistently is in the first iteration of EM, and the accuracy on the unseen test examples sharply degrades in the following iteration. These were the advertisement and business name data (Figures 3.8 and 3.3). For example, with 40% of the business name training data, naive Bayes achieves an accuracy of 19.5%. In the first EM iteration, this accuracy jumps to 26%; by the seventh iteration it fall to 23.3%. Since it is precisely these two data sets that have background knowledge that is ill matched to the classification classes as well as the training and test set data, as EM attempts to use the parameters of the classifier that were created with the background knowledge through more than one iteration the naive Bayes assumptions are severely violated.

3.4 Summary

We have substituted the use of background knowledge for unlabeled examples in an expectation maximization algorithm. Although at first glance this might seem to be counter-intuitive, we have shown empirically that even background knowledge that is not of the same form as the training data can provide information that allows the learner to improve accuracy on the test set.

Chapter 4

Incorporating Background Knowledge using WHIRL

In the previous chapter we approached the inclusion of background knowledge as simply adding a set of unlabeled examples to the text classification task. We then classified the background knowledge, and used them to enhance the model of the data. In this chapter we look at background knowledge from an entirely different angle. Our approach now views the task as one of information integration using WHIRL [11, 12], a tool that combines database functionalities with techniques from the information-retrieval literature. Unlike the preceding approach, we do not attempt to classify the background knowledge and indeed, definitely do not require that it be of a form comparable to that of the training data. We use the background knowledge as an extra corpus for the learner, to aid it in its decision task. Rather than directly comparing a new test example to elements of the labeled training corpus, we use the background knowledge as a "bridge", to connect the test example with labeled examples. A labeled training example is useful in classifying an unknown test instance if there exists some part of the unlabeled background knowledge that is similar to both the test example and the training example. We call this a "second-order" approach to classification, in that training and test data are no longer directly compared but, rather, are compared one step removed, through an intermediary.

A concrete example of the usefulness of our approach can be seen in the task of assigning topic labels to technical papers. Assuming a machine learning supervised model, we are given a corpus of titles of papers, each with an associated label. The only information available to the learner is contained in this labeled corpus. This labeled corpus might be insufficient or incomplete. For example, in labeling the title of a physics article with its sub-specialty, any title containing a word such as *galaxy* should easily be classified correctly as an astro physics paper, even if there are few training articles in that domain. This is the case because *galaxy* is an extremely common word that appears quite often in papers about astro physics. However, an article on a more unusual topic, as for example *old white dwarfs,* would only be classified correctly if a title with these words appears in the labeled training examples. Although the training set does not contain the words *old white dwarfs* in our experimental data, our system is able to correctly classify a title with these words as astrophysics, by utilizing a corpus of unlabeled paper abstracts from the same field, which is naturally available on the Web. In our second-order approach, our system finds those unlabeled paper abstracts that are most similar to both *old white dwarfs* and to various training titles. These training titles are then used to classify *old white dwarfs* correctly, although each of these titles is quite dissimilar to it when compared directly.

In order to achieve our goal, we use WHIRL [11, 12] which is a conventional database system augmented with special operators for text comparison. Words in text documents stored in the system are stemmed using Porter's stemming algorithm [52]. WHIRL is used as a nearest neighbor text classification program [14], with text documents specified as TFIDF vectors, and similarity between text documents measured as cosine similarity [57]. WHIRL makes it possible to pose SQL-like queries on databases with text-valued fields. We transform the training corpus, the test examples and the background knowledge into a database format and WHIRL provides a framework in which we can easily specify and explore second-order similarity classification. It allows for succinct queries that specify the combination of training similarity and background similarity to a new test example.

In the rest of this chapter we will give a review on WHIRL, and a discussion of how it is used for the text classification task. We will then provide the framework for our inclusion of background knowledge into the text classification task, including various options and design decisions. Result graphs on the same data sets and using the same methodology as those in Chapter 3 will then be presented.

4.1 WHIRL

WHIRL [11, 12] is an information integration tool that is specifically designed to query and integrate textual sources from the Web that do not necessarily conform to strict database conventions and formats.

4.1.1 Limitations of Conventional Databases

Assume that we have a database, *A* that has two fields, one which contains titles of papers and the other containing authors. A second database, *B* also has two fields, one that contains an author name and the other that contains the institution (company, university, etc.) that the author is affiliated with. These fields are all filled with free text, so that if a paper title has more than one author, the author field will contain a list of all the names. Suppose that we are searching for all papers where at least one of the authors is from *Rutgers, the State University of NJ.* A conventional database system with a SQL query capabilities would find this an impossible task. Although we wish to join on the field "author" from both databases, we cannot assume that these two fields use the same set of object identifiers.

As a simple straight-forward example, suppose database *A* has an entry:

Title	Author
Joins that Generalize: Text Categorization Using WHIRL	William Cohen and Haym Hirsh

and database *B* has entries:

Author	Affiliation
William W. Cohen	Whizbang! Labs, inc.
Haym Hirsh	Rutgers, the State University of NJ

The SQL query would ideally be:

```
SELECT A.title
  FROM A and B
 WHERE A.author = B.author
    AND B.affiliation = Rutgers, the State University of NJ
```

However, B.author is equal to *Haym Hirsh*, but A.author is equal to *William Cohen and Haym Hirsh*. Database systems cannot join on these two unequal values. Conventional databases deal with this problem by forcing the author fields to use the same set of object identifiers, and would not allow one author field to hold individual names and the other hold a sequence of names. However, when databases are filled in an automatic fashion based upon parsing Web pages, these restrictions are unrealistic. This makes the querying and joining of databases that have textual fields an almost impossible problem for database programs, without human intervention.

4.1.2 The Usefulness of WHIRL

WHIRL allows us to look at these fields as holding *text*, as opposed to holding specific values from a set of object identifiers. In WHIRL's terminology, the information we wish to store is placed in STIR (Simple Text in Relations) format, where fields can hold arbitrary textual values. WHIRL's approach to this problem allows these textual fields to be queried directly via new operators that are introduced into the SQL-type queries.

We can look at the example of a corpus of physics technical paper titles, labeled with subspecialties, that was mentioned in the introduction to this chapter. If we wish to place the corpus of physics paper titles (without authors) into a STIR relation, we can create a STIR relation named *Titles* with arity of 1, where each tuple of *Titles* contains one element that is a document vector, the title itself. Some elements in the relation (before stemming and TFIDF is computed) might therefore look as follows:

$\langle$ Asymmetry in Microlensing-Induced Light Curves $\rangle$

$\langle$ A Tully Fisher Relation for S0 Galaxies $\rangle$

$\langle$ Large Thermopower in a Layered Oxide NaCo$_2O_4\rangle$

It might be useful to hold more information than just the title of a paper in each of the entries of a relation. Suppose we are filling up this STIR relation from the physics paper e-print archives (http://xxx.lanl.gov), where each paper is placed into a specific area of physics and is specified by a title, authors, and some additional comments about the paper. STIR relations can hold conventional data types as well as vectors, so if we have a corpus of physics paper titles, with associated authors and a small set of comments about the paper, as well as the area of physics to which it is assigned, we can create a STIR relation called *Paper* of arity 4 where each tuple in *Paper* contains a vector holding the title, a vector holding the authors, a vector representing some small comments and a string that names the class of that particular title. The strings that specify the classes are predefined and can be joined and compared in conventional database manners. An example of some tuples in this new relation would be (actually taken from http://xxx.lanl.gov/list/):

$\langle$ The Impact of Cooling and Feedback on Disc Galaxies, Frank C. van den Bosch, 20 pages 12 figures To be published in MNRAS, astrophysics $\rangle$
$\langle$ An AGN Identification for 3EG J2006-2321, P. M. Wallace J. P. Halpern A. M. Magalhaes D. J. Thompson, 22 pages 6 figures To appear in ApJ v569 n1 10 April 2002, astrophysics $\rangle$

The STIR database can be accessed via an SQL-type language with queries of the form:

$$\begin{aligned} &\text{SELECT } \langle elements\rangle \\ &\text{FROM } \langle relations\rangle \\ &\text{WHERE } \langle conditions\rangle. \end{aligned}$$

In the above query $\langle elements\rangle$ and $\langle relations\rangle$ are defined to be the field names and the names of tables as in all SQL queries. $\langle conditions\rangle$ include the conditions that must be satisfied by the results that are returned and these conditions are also the same as in SQL queries with one additional operator allowed. WHIRL introduces the operator SIM which is used to compare elements that are document vectors. Instead of assuming that keys must be exactly identical in order for relations to be joined, this introduces the concept of a "soft join", which pairs elements with similar values, rather than equal ones. When the SIM operator is used, distances between vectors are computed using the cosine metric, which represents the statistical similarity between documents. An element that is a document, x_j, is represented as a vector of terms weights $\langle w_{j1}, \ldots, w_{j|T|}\rangle$, where T is the set of terms in all the documents. These weights are generated using the standard TFIDF method [57], where w_{jt} equals $log(TF_{x_jt}) \times log(IDF_t)$. TF_{x_jt} corresponds to the number of times that term t occurs in document x_j and IDF is the total number of documents divided by the number of documents that contain the term t. If two elements that are document vectors, x_i and x_j are compared via the SIM operator the value of the score that is computed is $\sum_{t \in T} x_{it} \times x_{jt}$. Since all vectors are made to be of unit length this value is always between 0 and 1. WHIRL allows the user to specify a parameter k, that indicates how many documents are to be returned in answer to a query. Those k documents with the highest score are returned.

Suppose that we have a large corpus of documents, and we wish to find some k documents that are most similar to a specific document d. This is an information retrieval problem that can be addressed by WHIRL. Suppose we have a new physics paper title (the document d in this case) and we wish to find titles that we have stored (the large corpus of documents) that share one or more of the authors of this new paper. We can use WHIRL in the following manner:

- Create a new relation, *New*, that has one element containing the new title and new author

- Query the relation *New* and the relation *Paper* with an SQL type query

```
SELECT Paper.title
  FROM New AND Paper
WHERE New.author SIM Paper.author
```

This query will return all those titles from the relation *Paper* that have an author value that is similar to the author value of the new title that we have obtained. Unlike conventional SQL, we cannot say that the tuples returned by the WHIRL system make the query "true" in the Datalog sense of substitution of variables, but rather, each tuple returned has a score associated with it to tell us how similar the clause in the WHERE is. If the new paper shares all authors (and all are spelled the same way, with the same initializations) with a paper in *Paper* then this score will be 1; as fewer authors are shared the score approaches 0.

4.2 WHIRL for Text Classification

Assume that we have a corpus of training examples that are free text with labels, and a test example that must be assigned a label. The training examples can be viewed as a table with the field *instance*, to hold the textual data, and field *label* to hold the class label. The test example is a one line table, with simply the textual field *instance*. An example of a WHIRL query [14] is:

```
SELECT Test.instance, Train.label
  FROM Train AND Test
 WHERE Train.instance SIM Test.instance
```

Given a user-specified parameter k, this query will first generate an intermediate table containing the k tuples

$$\langle \textit{Test.instance}, \textit{Train.instance}, \textit{Train.label} \rangle$$

that maximize the similarity score between *Test.instance* and *Train.instance*. Unlike traditional SQL queries, the result of this is a set of tuples ordered by score, with the highest score representing the closest *Train.instance*, *Test.instance* pair, using WHIRL's SIM operator to compute the similarity of these textual fields. An example of this intermediate table can be seen in Figure 4.1. Here the columns correspond to the score, the test example which was input using standard input, the label of the training example that is close to the test example, and the training example that is close to the test example (written as a file name and offset). This example comes from the NetVet domain.

```
score       test       label    train

0.758169    -1+stdin   horse    12172+train.data
0.569772    -1+stdin   horse    8485+train.data
0.569772    -1+stdin   horse    8654+train.data
0.569772    -1+stdin   horse    8700+train.data
0.516987    -1+stdin   horse    3693+train.data
0.514894    -1+stdin   horse    12067+train.data
0.50018     -1+stdin   horse    2982+train.data
0.50018     -1+stdin   cow      40573+train.data
0.50018     -1+stdin   dog      75432+train.data
0.498727    -1+stdin   dog      62567+train.data
0.498727    -1+stdin   cat      78135+train.data
0.471873    -1+stdin   horse    4815+train.data
0.471061    -1+stdin   cow      43540+train.data
0.411263    -1+stdin   rodent   32151+train.data
0.408616    -1+stdin   cat      86408+train.data
0.408616    -1+stdin   dog      62445+train.data
```

Figure 4.1: Example of results returned by a WHIRL query

In WHIRL's final step it takes this table of k tuples and projects it onto the fields specified in the SELECT statement. Note that this can mean that there may be many training examples among the k with the same label (i.e., multiple nearby examples in the training set), and these are combined into a single tuple in the final result table. The combination of scores is performed by treating scores as probability and the combination as a "noisy or." If the individual scores of the tuples with a given label are $\{s_1, \ldots, s_n\}$, the final score for that label is $1 - \prod_{i=1}^{n}(1 - s_i)$. Whichever label has the highest score in the resulting projected table is returned as the label for the test instance. This method bears many similarities to the nearest-neighbor method of [63], which has been shown to perform quite well on text classification tasks [14]. Indeed, based on these two papers we also use a value of $k = 30$ in our experiments. An example of a projection table that uses the score table that is shown in Figure 4.1 (but uses the top 30 results) can be seen in Figure 4.2. In this table, there is only one line per label, and the score for each label is the combination of scores from the larger table.

n	score	class
0	0.999901	horse
1	0.929449	dog
2	0.839594	cow
3	0.824687	cat
4	0.573973	bird
5	0.411263	rodent
6	0.272381	primate

Figure 4.2: Example of a projection as done by WHIRL

4.3 Comparison of WHIRL for Text Classification with Other Methods

Before we inject background knowledge into this text classification process, because a number of the data sets are being used for the first time in this body of research, we start by comparing the core WHIRL method for text classification without background knowledge (which we label WHIRL-nn), to some other traditional methods for these types of problems. We present accuracy results for WHIRL-nn, RIPPER [10], and naive Bayes. Accuracy rates in Table 4.1 represent average accuracy of five cross-validated runs on the full training set. Since WHIRL uses the Porter's stemming algorithm when it creates the TFIDF representation, we used it as well, before giving the data to the other learners. As can be seen from this table, WHIRL-nn is comparable to other methods, and thus any improvements above and beyond WHIRL-nn that we now report represent even stronger classification performance than this credible state-of-the-art

Table 4.1: Accuracy: Comparison of Text Learning Algorithms

Data Set	RIPPER	WHIRL-nn	Naive Bayes
20-News	28.5	**60.7**	55.5
WebKb	82.3	79.5	**88.8**
advertisements	62.1	**90.1**	88.8
ASRS	**64.5**	64.4	60.6
2class physics	69.5	91.2	**94.3**
3class physics	60.0	86.2	**88.3**
NetVet	55.7	**60.8**	59.7
Business Names	22.1	**30.1**	23.5
Clarinet	88.3	95.2	**95.6**
thesaurus	24.5	**47.9**	35.6

method.

4.4 WHIRL with Background Knowledge

A WHIRL query can have an arbitrary conjunction in its WHERE clause. The final score for a returned tuple is the product of all the scores of all the individual components of the WHERE clause. In this way, each of these component scores can be viewed as independent probabilities that are combined to produce a final probability that the returned tuple accurately answers the query. This gives a great amount of flexibility in the formulation of queries and the addition of alternative tables, or sources of knowledge, into the queries.

Such alternative sources of knowledge may provide us with a corpus of text that contains information both about importance of words (in terms of their TFIDF values in this large corpus), and joint probability of words (what percentage of the time do two words coexist in a document?). This gives us a large context in which to test the similarity of a training example with a new test example. We can use this context in conjunction with the training examples to label a new example.

Because of WHIRL's expressive language, and the ability to create conjunctive queries simply by adding conditions to a query, WHIRL's queries for text classification can be expanded to allow for the use of background knowledge on a subject. In the example of the classification of physics paper titles discussed earlier, suppose that we had a fairly small set of labeled paper titles, and also a very large set of unlabeled titles, papers or abstracts (or Web pages resulting from a search), in a relation called Background with a single field, *value*. We can create the following query for classification:

```
SELECT Test.instance, Train.label
  FROM Train AND Test AND Background
WHERE Train.instance SIM Background.value
   AND Test.instance SIM Background.value
```

Given a query of this form WHIRL will first find the set of k tuples $\langle X_i, Y_j, Z_k, L_j \rangle$ from the Cartesian product of *Train* and *Test* and *Background* such that $SIM(Y_j, Z_k) \times SIM(X_i, Z_k)$ is maximal. Here each of the two similarity comparisons in the query computes a score, and WHIRL multiplies them together to obtain the final score for each tuple in the intermediate-results table. The intermediate results table has the elements from all three of the tables that are in the FROM statement and the score:

$$\langle Test.instance, Train.label, train.instance, Background.value, score \rangle.$$

This table is then projected onto the *Test.instance* and *Train.label* fields as discussed before. Whichever label gives the highest score is returned as the label for the test example.

One way of thinking about this is that rather than trying to connect a test example directly with each training example, it instead tries to bridge them through the use of an element of the background table. Note that WHIRL combines the scores of tuples generated from different matches to the background table. A schematic view of this can be seen in Figure 4.3. The rectangles in Figure 4.3 represent individual documents in each of the corpora. If a line connects two documents then it represents the fact that those two documents are similar using the cosine similarity metric. If a path can be followed from the test document to a training document via a piece of background knowledge then then those two documents are considered to be similar in our scheme. Our use of WHIRL in this fashion thus essentially conducts a search for a set of items in the background knowledge that are close neighbors of the test example, provided that there exists a training example that is a neighbor of the background knowledge as well. As can be seen from Figure 4.3, training examples can be used multiple times with different background knowledge and a piece of background knowledge can be used multiple times as well, with different training examples.Training neighbors of a test example are defined differently when background knowledge is incorporated. If words in a test example are found in some background knowledge, then other words that are in that background knowledge can connect this test example to dissimilar (in terms of word overlap and direct cosine difference) training examples. The final classification thus integrates information from multiple training examples and the multiple "bridge" examples that lie between them in the background text.

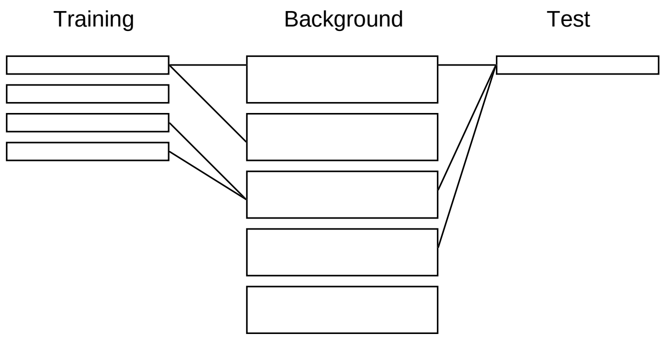


Figure 4.3: Schematic view of WHIRL with background knowledge

Note that this approach does not concern itself with which class (if any!) a background item belongs to. We instead simply use the text directly as part of the decision-making process. This is in contrast to the approach discussed in Chapter 3 that explicitly uses the training set to classify the background items as if they were true examples, and then adds them to the labeled set. Our method allows for more sophisticated use and combination of the training instances and background knowledge. A background instance that is close to numerous training instances can be included more than once in the table returned by the WHIRL query – even if the training examples that it is close to have different classes. Similarly, a training example can also be included in the table multiple times, if it is close to numerous background instances. Suppose that our classification task consists of labeling the first few words of a news article with a topic. If a test example belongs to the category *sports*, for instance, the cosine distance between the few words in the test example and each of the small number of training examples might be large. However, we would hope that given a large corpus of unlabeled news articles, it is likely that there will be one or more articles that contains both the few words of the test example and the words of one of the training examples.

To make our classification system more robust, if the background query that we presented does not provide a classification label for a given test example, we then allow the simple text classification query (WHIRL-nn) to attempt to classify the test example. If this query fails as well, then the majority class is chosen as a final attempt at classification. Consider the test example in the NetVet domain:

Steller's eider (USFW)

USFW is an acronym for the U.S. Fish and Wild Life Service, which is not referred to this way in the background corpus. The word *eider*, which is a type of duck does not appear in the background corpus either. WHIRL-bg therefore does not return anything. However, the training set contains the example:

Migratory Birds and Waterfowl – USFWS

so once WHIRL-nn is used, the correct class is returned.

We term this method of using background knowledge for the text classification task, WHIRL-bg.

4.4.1 Results for WHIRL-nn and WHIRL-bg

The accuracy results for the 20 Newsgroups data, the WebKb data, the advertisements data, and the ASRS data are graphed in Figures 4.4–4.7. The y axis represents the accuracy rate, and the x axis is the number of labeled examples given to the learner. In the first three of these graphs the same phenomenon is apparent: with few labeled examples per class the background knowledge is very helpful, but as the number of labeled examples increase, the usefulness of the background knowledge goes down, and even causes the accuracy to degrade. In the ASRS data, background knowledge helps most when there are more than 2 examples per class, because it is sometimes the case that the *very* small training sets do not provide enough information (in terms of vocabulary in each class) to utilize the background knowledge fully.

We present two sets of results on the physics data in Figure 4.8 and Figure 4.9. Figure 4.8 is a two-class problem, where only the titles of papers in the astrophysics and condensed materials classes were used. These classes had nearly the same number of training examples. Figure 4.9 is a three class problem, where a class with a fewer number of training examples was added to the previous problem. Figure 4.8 clearly shows the effect that background knowledge can have on text data sets. The line representing WHIRL-bg remains almost horizontal as fewer training examples were used, indicating that the background knowledge compensated for the lack of data. In contrast, WHIRL-nn sharply degraded as fewer training examples are used. The helpfulness of the background knowledge, therefore, also increased as fewer training examples were used. When the third class was added, error rates of both WHIRL-nn and WHIRL-bg went up. However, the same effect of background knowledge can be seen in Figure 4.9 as well.

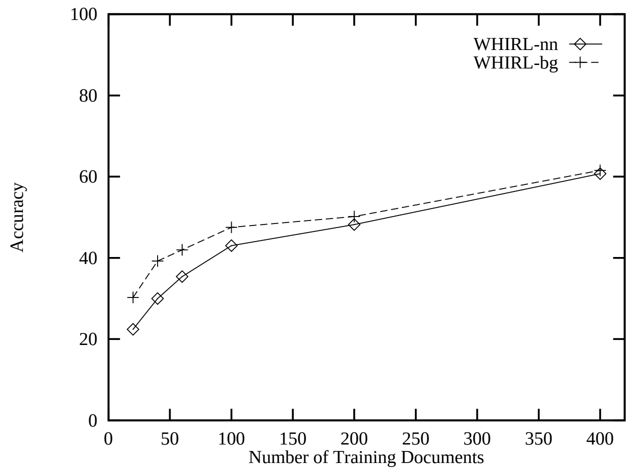


Figure 4.4: WHIRL-nn and WHIRL-bg for 20 Newsgroups

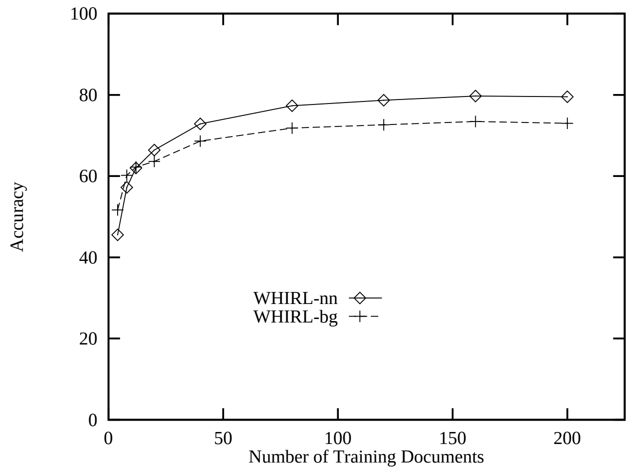


Figure 4.5: WHIRL-nn and WHIRL-bg for WebKb

Results for the NetVet domain are graphed in Figure 4.10. Reductions in error rate increased

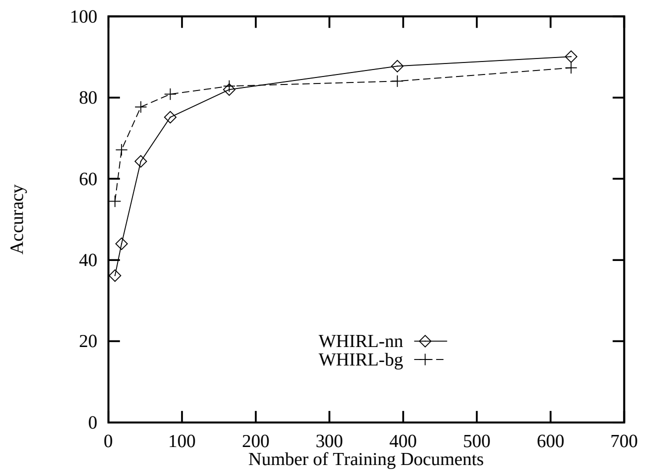


Figure 4.6: WHIRL-nn and WHIRL-bg for the advertisements problem

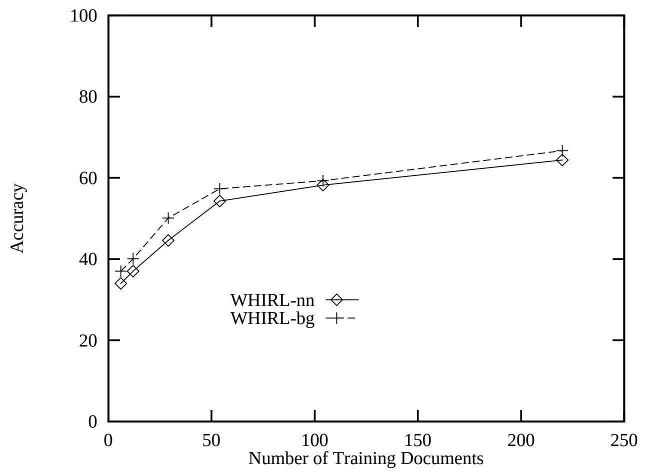


Figure 4.7: WHIRL-nn and WHIRL-bg for the ASRS problem

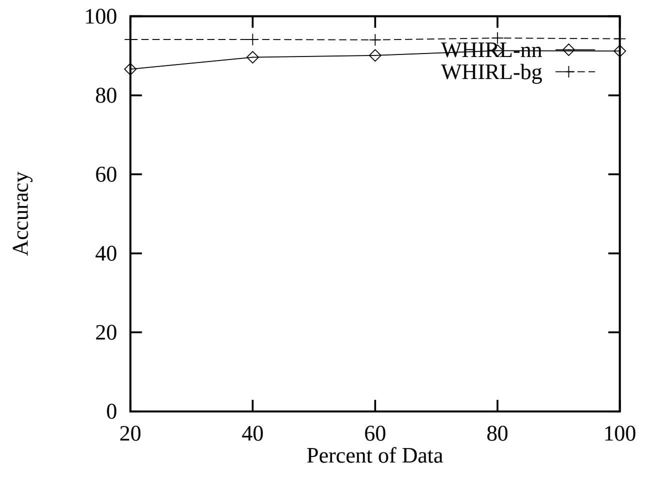


Figure 4.8: WHIRL-nn and WHIRL-bg for the 2-class physics title problem

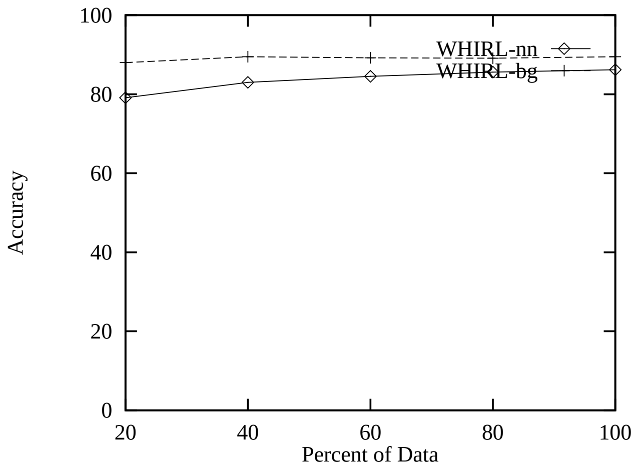


Figure 4.9: WHIRL-nn and WHIRL-bg for the 3-class physics title problem

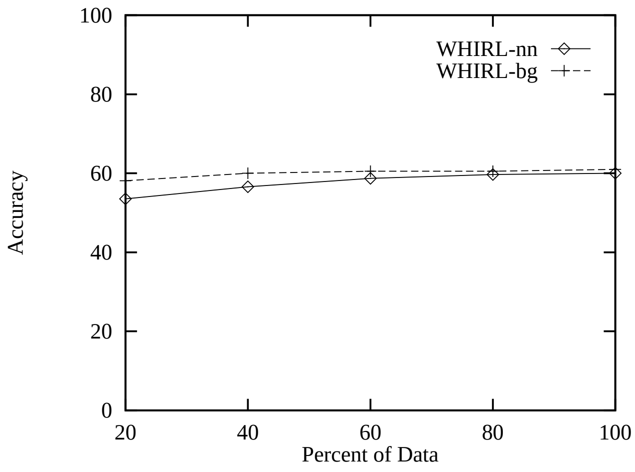


Figure 4.10: WHIRL-nn and WHIRL-bg for the NetVet data

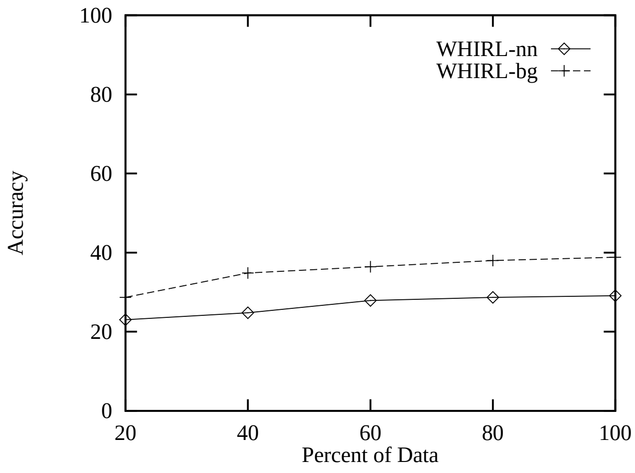


Figure 4.11: WHIRL-nn and WHIRL-bg for the business names data

as the number of training examples decreased. The NetVet domain is unlike some of the other sets previously discussed in that there was overlap in topics in the background knowledge. A Web page that could be useful in classifying a test example as belonging to the category of Dogs was quite likely to discuss Cats and vice versa. Some of the training and test examples, too, could have caused confusion. There were titles of Web pages on pet stores or animal care that were placed under one topic, but could just have easily been placed in many other different categories. We therefore were not surprised to see that the error rate did not decrease by a large percentage.

The results for the business name data set are graphed in Figure 4.11. Once again, WHIRL-bg outperformed WHIRL-nn. Using 100 percent of the data, the decrease in error rate is substantial. However, when the percent of training examples that was used is lower, the difference in error rate between the two systems is reduced. This is unlike the results of the previous three domains. This might have been due to the fact that the training and test examples were company names, which often consisted of words that occured only once (for example, *Xerox*) so that reducing the number of training examples actually reduced the dictionary of words in the training corpus substantially. There were therefore fewer words that could be used to find bridges in the background knowledge. This same trend can be seen in Figure 4.13, which graphs accuracy on the thesaurus data. Here too, the labeled data consists of single words, so with only 20% of the data, the usefulness of background knowledge is not as apparent as when more labeled data is added, although there are still substantial improvements in accuracy rates.

Results for the Clarinet news problem is in Figure 4.12. The addition of background knowledge was useful when the training set size was small. When less than 60% of the data was used, background knowledge reduced the error rate. As the amount of training data increased the background knowledge no longer added enough new vocabulary to improve performance.

Once again we wished to determine whether the difference in accuracy obtained using WHIRL vs using WHIRL-bg is statistically significant. From Figures 4.4–4.13 we can obtain 72 pairs of numbers. For each x value that is plotted on one of these graphs, the corresponding two y values create a pair of numbers. The first number in each pair represents the accuracy of running WHIRL while the second number is the accuracy of running WHIRL-bg on the same set. We used a paired t-test on these 72 pairs of numbers to determine if the addition of background

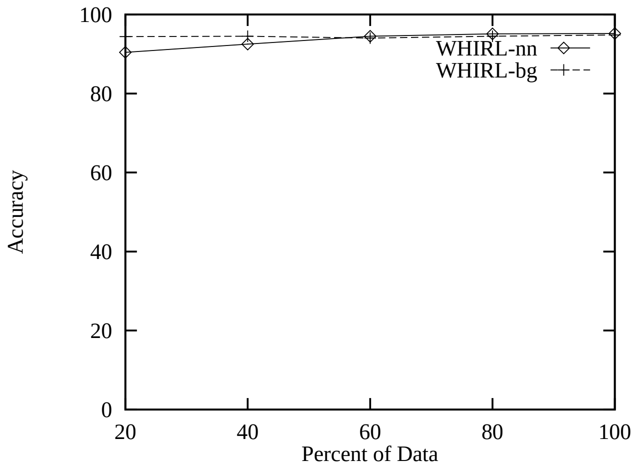


Figure 4.12: WHIRL-nn and WHIRL-bg for the Clarinet problem

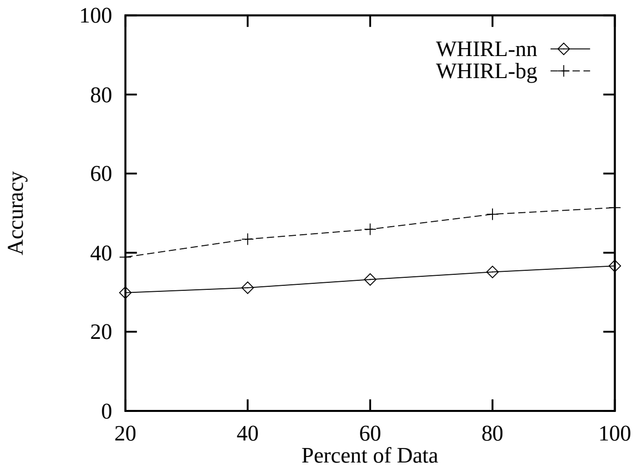


Figure 4.13: WHIRL-nn and WHIRL-bg for the thesaurus problem

knowledge caused a significant improvement in the accuracies. Testing the hypothesis that the difference in accuracies was simply the result of chance, the p value was less than .01, and hence we were able to conclude that significant improvements in accuracy are expected if background knowledge is added to the WHIRL query.

4.4.2 Unrelated Background Knowledge

It is still the case that the *conjunctive* query that we presented incorporates background knowledge in a way that overlooks the direct comparison between training and test examples. Depending upon the type of data, and the strength of the background knowledge, this might be a dangerous approach. One of the strengths of WHIRL as a data retrieval engine is that if the test example exists in the training corpus, and the SIM function compares the test example to the training examples, the training example that is identical to the test example will be returned with a score equal to one. Our "second order" approach weakens WHIRL so that this quality is no longer true. If the conjunctive background query returns a set of results then the test example is never directly compared to the training examples and we can no longer access their direct similarity. If a training example is identical to the test example, but is not close to any element in the background knowledge database, it is possible that it will not even be returned in the top k results of the intermediate table. We wish to minimize the risk of such an anomalous event occurring. Additionally, if the background knowledge that is used is unrelated to the text classification task, WHIRL-bg can degrade drastically. Consider the ridiculous situation of the NetVet classification task using the background knowledge from the physics data set; i.e. a set of technical paper abstracts. If a test example consisting of a veterinary web page titles cannot be compared to any of the abstracts then the background query will return nothing, and the system will fall through to WHIRL-nn to classify the instance. However, suppose that a test example *can* be compared to an abstract, as meaningless as the comparison might be. We would then have very misleading results. The test example:

russian horses in the UK

when run in this case returns as a result:

n score predicted label

Table 4.2: Comparison of accuracies of WHIRL-nn and WHIRL-bg with unrelated background knowledge

Data Set	WHIRL-nn	WHIRL-bg
NetVet-20	53.5	36.5
NetVet-40	56.6	39.0
NetVet-60	58.7	40.3
NetVet-80	59.7	40.0
NetVet-100	60	40.9

```
0 0.0464414 dog 1 0.0434904 cow 2 0.0333697 rodent 3 0.0171406 bird 4 0.0145024
primate 5 0.0142586 horse
```

The scores are extremely low, since the background knowledge is not close to either the train or the test set. This example, as expected, is misclassified as *dog*. If we run the NetVet data using this background knowledge from the physics domain, WHIRL-bg performs much worse than WHIRL-nn, as can be seen from Table 4.2. Table 4.2 gives the average accuracy rates for 20, 40, 60, 80, and 100% of the NetVet data using WHIRL-nn, and using WHIRL-bg with the incorrect background knowledge.

We wish to make our system more robust to the inclusion of misleading background knowledge. To do this we create a *disjunctive* query that combines both the standard text classification using WHIRL with the WHIRL-bg approach.

```
SELECT Test.instance as instance
    AND Train.label as predicted
  FROM Train AND Test
 WHERE Train.instance SIM Test.instance

SELECT Test.instance as instance
    AND Train.label as predicted
  FROM Train AND Test AND Background
 WHERE Train.instance SIM Background.value
    AND Test.instance SIM Background.value
```

Using the two queries that we presented above we can create intermediate tables of their results separately, and project onto the test *instance* and *label* fields separately as well. These two sets of results are then combined by defining a disjunctive view. This query selects a test *instance* and *label* from the results of WHIRL-nn and also selects a test *instance* and *label* from the WHIRL-bg query. (see Figure 4.14). Figure 4.14 defines three separate views using the *inview* command in WHIRL. These three views are: *label view WHIRL-nn*, *label view WHIRL-bg*, and *disjunction*. The view *disjunction* is first made empty with the *clear* command, and then is defined to be filled with the two previous *inview* commands to include the results from WHIRL-nn and WHIRL-bg. Once these three views are defined, the *materialize* command is used to actually fill the views with the data from the *train*, *test*, and *back* data.

When this disjunctive query is materialized (see Figure 4.15) once again there may be multiple lines with the same label but with different scores. The noisy or is then used on this combined table to arrive at a final result. This is equivalent to producing the two intermediate tables of size k and applying the noisy or to all the results returned by those two tables. If either of the two queries returns elements with very high scores, then those will dominate the noisy or operation. In empirical testing we have found that this query is comparative to the WHIRL-bg query that is defined above; it improves upon learning without background knowledge. The main advantage of this query is that when the background knowledge is misleading and returns results that are meaningless, the disjunction prevents the system from placing emphasis on these false comparisons. The test example:

russian horses in the UK

with the disjunctive query, returns a final results of:

n score predicted label

0 0.997777 horse 1 0.76109 dog 2 0.521918 cow 3 0.0333697 rodent 4 0.0171406 bird 5 0.0145024 primate

This is a combination of the results from WHIRL-bg (that was given above) and the results from WHIRL-nn which were:

n score predicted label 0 0.997745 horse 1 0.749454 dog 2 0.50018 cow

Since the scores returned in the comparisons of WHIRL-nn were much larger, they dominate the result, which is exactly what we would like. Table 4.3 shows a comparison between the

```
inview  label_view_WHIRL-nn
from    test
and     train
where   train.instance sim test
select   test.VAL as test
and     train.label as label
end

inview  label_view_WHIRL-bg
from    test
and     train
and      back
where   back.value sim test
and     back.value sim train.instance
select   test.VAL as test
and     train.label as label
end

clear disjunction
inview disjunction
select  WHIRL-bg.test
and    WHIRL-bg.label
from    WHIRL-bg
end

inview disjunction
select  WHIRL-nn.test
and    WHIRL-nn.label
from   WHIRL-nn
end
```

Figure 4.14: Specification of views in WHIRL for the disjunctive query

```
materialize label_view_WHIRL-nn as WHIRL-nn
materialize label_view_WHIRL-bg as WHIRL-bg
materialize disjunction as final_answer
browse final_answer 0 30
```

Figure 4.15: Materialization of view in WHIRL for the disjunctive query

Table 4.3: Comparison of accuracies of WHIRL-nn and WHIRL-dis with unrelated background knowledge

Data Set	WHIRL-nn	WHIRL-dis
NetVet-20	53.5	51.5
NetVet-40	56.6	54.0
NetVet-60	58.7	57.1
NetVet-80	59.7	57.7
NetVet-100	60	57.8

accuracy rates of WHIRL-nn and WHIRL-dis with unrelated background knowledge. As we can see, the unrelated knowledge still hurts somewhat with WHIRL-dis, but not nearly as much as with WHIRL-bg (see Table 4.2).

In cases where the background knowledge is very related to the problem, especially if the data consists of very short text strings, WHIRL-bg may be more useful than WHIRL-dis. This is partly because the direct comparison part of the disjunctive query has only one conjunct, whereas the background part of the query has two conjuncts. Since WHIRL multiplies the scores of the two conjuncts, and these scores are less than one, the background part of the query often has scores that are lower than the direct comparison part of the query. This reduces the utility of the background knowledge when the two parts of the disjunctive query is combined. This phenomenon can be observed in the business name data set, where WHIRL-bg outperforms WHIRL-dis (see Table 4.4). The direct comparison of WHIRL-dis often relies on words such as *inc* or *company* that are not very informative, yet often provide higher scores than the background part of the WHIRL query, since the score of the background is the product of two conjuncts. WHIRL-dis would therefore be a better choice if the background knowledge is not from a reliable source, and is not known to be closely related to the problem; otherwise, WHIRL-bg would be the appropriate choice.

We present a table (Table 4.5) with four of the data-sets that we described in Chapter 2, to illustrate how the disjunctive query performs in the presence of misleading background knowledge. The four data sets are the 2-class physics paper title problem, the NetVet problem, the thesaurus words problem, and the business name problem. We choose these four problems each

Table 4.4: Comparison of accuracy of WHIRL-bg and WHIRL-dis with correct background knowledge

Data Set-percent	WHIRL-bg	WHIRL-dis
Business-20	35.6	23.0
Business-40	38.3	26.0
Business-60	39.4	27.7
Business-80	40.2	29.5
Business-100	40.3	29.3

time that we do further studies on our basic results, because each has a different type of background knowledge. The physics data has background knowledge that is of the same type as the data, and the NetVet background knowledge is from the same domain as the data but of a slightly different type. The thesaurus background knowledge is very different than the data but each piece is only about one word, which is like the training and test set, and the business data has background of a totally different size and type than the training and test set.

Each result that is presented in (Table 4.5) is once again the average of five-fold cross validation, and we show results for 20, 40, 60, 80, and 100% of the data. For each of these classification problems we present the accuracy rates of the WHIRL-bg query and the disjunctive query, which we will term WHIRL-dis. For the WHIRL-bg query and the disjunctive query we present results for the background knowledge that is related to the task (as described in Chapter 2), for background knowledge that is a combination of the one related to the task as well as one unrelated (under the column heading *mixed* in the table), and for totally unrelated background knowledge (under the column heading *wrong* in the table). For the unrelated background knowledge we use the background set from the NetVet data for the other three tasks, and the physics abstracts for the NetVet task. The mixed background set consists of all documents in the related background set plus all documents in the unrelated set of background knowledge for each task.

In all four data sets, for any number of training examples, there is a major discrepancy in the accuracy rate of WHIRL-bg with the wrong set of background knowledge and WHIRL-dis with the wrong set of background knowledge. In many cases, WHIRL-bg with the wrong set of background knowledge had accuracy rates that were substantially lower even than WHIRL-nn, without any background knowledge. For example, with the Physics data set, using 20% of

Table 4.5: Accuracy: Comparisons of WHIRL-bg and WHIRL-dis with variations of background knowledge

Data Set percent	WHIRL-bg	WHIRL-bg mixed	WHIRL-bg wrong	WHIRL-dis	WHIRL-dis mixed	WHIRL-dis wrong
thesaurus-20	**38.9**	37.3	24.6	38.7	36.5	24.8
thesaurus-40	43.4	42.1	26.1	**44.1**	43.1	29.2
thesaurus-60	45.9	46.5	26.2	**47.6**	47.4	31.4
thesaurus-80	48.7	48.6	27.5	50.5	**51**	34.1
thesaurus-100	51.4	51.7	26.8	53.0	**53.1**	35.6
NetVet-20	**58.1**	**58.1**	36.5	56.4	56.2	51.5
NetVet-40	60.0	**61.2**	39.0	59.5	59.6	54
NetVet-60	60.5	**62**	40.3	61.1	60.0	57.1
NetVet-80	60.5	**61**	40.0	61.1	62.4	57.7
NetVet-100	61.0	**61.9**	40.9	61.1	61.5	57.8
Physics-20	**94.1**	**94.1**	69.3	91.2	91.7	86.6
Physics-40	94.1	**95**	71.4	92.1	92.5	90.1
Physics-60	94.0	**94.5**	73.5	91.6	91.8	90.5
Physics-80	94.5	**95.1**	71.8	93.1	93.2	91.5
Physics-100	94.3	**95**	73.7	94.3	94.2	93.7
Business-20	**35.6**	25.8	15.9	23.0	23.4	22.2
Business-40	**38.3**	27.3	16.7	26.0	25.8	25.6
Business-60	**39.4**	28.6	17.4	27.7	27.2	27.1
Business-80	**40.2**	29.5	17.8	29.5	28.9	29.3
Business-100	**40.3**	28.7	18.1	29.3	29.5	30.1

the data, WHIRL-nn has an accuracy rate of 86.6%. WHIRL-bg achieves 94.1% accuracy with the correct background knowledge as well as the mixed background knowledge; WHIRL-dis achieves 91.2% accuracy for the correct and mixed background knowledge. For the wrong set of background knowledge, the accuracy for WHIRL-dis is the same as for WHIRL-nn, which is what we would like to occur. The inappropriate background knowledge does not help learning, but does not hurt it either. However, for WHIRL-bg with the wrong set of background knowledge, accuracy actually degrades sharply, achieving a level of only 69%. This phenomenon can be seen in all the data sets, so we are convinced that WHIRL-dis minimizes the effects of misleading background knowledge. In the first three data sets presented, WHIRL-dis and WHIRL-bg perform similarly for the correct set of background knowledge; for the business name data set, WHIRL-bg outperforms WHIRL-dis.

4.5 Summary

We have presented a method to reduce error rates in text classification by using a large body of potentially uncoordinated background knowledge. In most of the data sets to which the system was applied, we saw substantial reductions in error rates, particularly when the set of labeled examples was small. In many cases the use of background knowledge allowed for only a small degradation in accuracy as the number of training examples was decreased. We presented two different queries, WHIRL-bg and WHIRL-dis. WHIRL-bg would be the method of choice if the background knowledge is very closely related to the data, otherwise WHIRL-dis would be preferable.

Chapter 5

Using LSI with Background Knowledge

In Chapter 3 we discussed using background knowledge as a corpus of *unlabeled examples* that is added to the text classification task. Each element of the background knowledge was then labeled and used as part of the training set to increase the accuracy of a naive Bayes model that that the learner was creating. In Chapter 4 the background knowledge was looked at as an *independent corpus of information* that was used to associate a test example with a set of examples from the training data. The classification of the test data in Chapter 4 used a nearest neighbor approach, with the definition of a training example being a close "neighbor" of the test example changing as the specification of the WHIRL query for classification changed.

In this chapter background knowledge is incorporated into the text classification task in a new way. The system that we describe in this chapter neither classifies the background knowledge, nor does it *directly* compare it to any training or test examples. Instead, it exploits the fact that knowing that certain words often co-occur may be helpful in learning, and that these word co-occurrences could be discovered from large collections of text in the domain. Therefore, the background knowledge is looked at as a large corpus of related data that will give clues to and help find semantic concepts in our training data.

To find these concepts that are helpful to learning we use Latent Semantic Indexing (LSI) [18, 21, 22, 23, 24, 25]. LSI is an automatic method that redescribes textual data in a new smaller semantic space. LSI assumes that there exists some inherent semantic structure between documents and terms in a corpus. The new space that is created by LSI places documents that appear related semantically in close proximity to each other. LSI is believed to be especially useful in combating polysemy (one word can have different meanings) and synonymy (different words are used to describe the same concept), which can make text classification tasks more difficult.

The key idea in this chapter is that we use the background text in the creation of this new re-description of the data, rather than relying solely on the training data to do so.

We once again approach the classification task using the nearest neighbor paradigm. As in all nearest neighbor approaches, in order to classify a test example, the distance between the training examples and the test example is computed, and the label of the nearest neighbor, or a vote among the labels of the k nearest neighbors determines the final classification label. Instead of directly comparing a test example to the training data, we first redescribe the space of training examples, using the background knowledge to help in this redescription, and then subsequently redescribe the test example in this same new space. The comparison between the training examples and the test example is then done in this smaller space. The background knowledge is in a sense used in a preprocessing phase of the data, as opposed to during the actual comparisons for classification. The background knowledge is used to create a new semantic space, and then this new space, but not the actual background examples, is used during classification.

In the next section we give a review of LSI, and describe how we use it for traditional text classification as well as for classification in the presence of background text. We then present and describe the results of the system on the different data sets, both with and without background knowledge.

5.1 Latent Semantic Indexing

Latent Semantic Indexing [22, 21, 23, 18] is based upon the assumption that there is an underlying semantic structure in textual data, and that the relationship between terms and documents can be redescribed in this semantic structure form. Textual documents are represented as vectors in a vector space. Each position in a vector represents a term (typically a word), with the value of a position *i* equal to 0 if the term does not appear in the document, and having a positive value otherwise. Based upon previous research [21] we represent the positive values as a local weight of the term in this document multiplied by a global weight of the term in the entire corpus. The local weight of a term *t* in a document *d* is based upon the log of the total frequency of *t* in *d*. The global weight of a term is the entropy of that term in the corpus, and is therefore based upon the number of occurrences of this term in each document. The entropy equals $1 - \sum_d \frac{p_{td} log(p_{td})}{log(n)}$

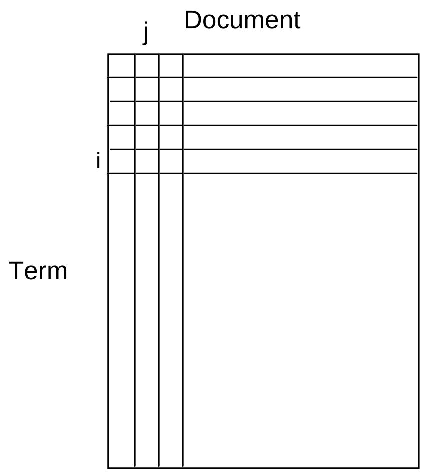


Figure 5.1: A term by document matrix

where n is the number of documents and p_{td} equals the number of times that t occurs in d divided by the number of total number of times that t occurs. This formula gives higher weights to distinctive terms. Once the weight of each term in every document is computed we can look at the corpus as a large term-by-document ($t \times d$) matrix X, with each position x_{ij} corresponding to the absence or weighted presence of a term (a row i) in a document (a column j) (see Figure 5.1). This matrix is typically very sparse, as most documents contain only a small percentage of the total number of terms seen in the full collection of documents.

Unfortunately, in this very large space many documents that are related to each other semantically might not share any words and thus appear very distant, and occasionally documents that are not related to each other might share common words and thus appear to be closer than they actually are. This is due to the nature of text, where the same concept can be represented by many different words, and words can have ambiguous meanings. LSI reduces this large space to one that hopefully captures the true relationships between documents. To do this, LSI uses the singular value decomposition of the term by document ($t \times d$) matrix.

The singular value decomposition (SVD) of the $t \times d$ matrix, X, is the product of three matrices: TSD^T, where T and D are the matrices of the left and right singular vectors and S is the diagonal matrix of singular values. The diagonal elements of S are ordered by magnitude, and therefore these matrices can be simplified by setting the smallest k values in S to zero.[1]

[1]The choice of the parameter k can be very important. Previous work has shown that a small number of factors

The columns of T and D that correspond to the values of S that were set to zero are deleted. The new product of these simplified three matrices is a matrix $\hat{X}$ that is an approximation of the term-by-document matrix. This new matrix represents the original relationships as a set of orthogonal factors. We can think of these factors as combining meanings of different terms and documents; documents are then re-expressed using these factors.

5.1.1 LSI for Retrieval

When LSI is used for retrieval, a query is represented in the same new small space that the document collection is represented in. This is done by multiplying the transpose of the term vector of the query with matrices T and S^{-1}. Once the query is represented this way, the distance between the query and documents can be computed using the cosine metric, which represents a numerical similarity measurement between documents. LSI returns the distance between the query and all documents in the collection. Those documents that have higher cosine distance value than some cutoff point can be returned as relevant to the query.

5.1.2 LSI for Classification

We are using LSI for text *classification*, so we can henceforth refer to the document collection as the training examples and the query as a test example. LSI allows the nearest neighbors of a test example in the new space to be returned, even if the test example does not share any of the raw terms with those nearest neighbors. As a simple example, we can look at one run of the WebKb data, where the training set and test set both consist of Web pages from universities. In the example we present the training set has only one example per class for a total of 4 examples: one from each of the classes *course, student*, *project*, and *faculty*. With this small training set, using the test set discussed in Chapter 2 of 225 examples, a state-of-the-art nearest neighbor program achieves 37% accuracy; LSI achieves 40% accuracy. The example Web pages presented below have been pre processed to remove all HTML, and to remove all words except the 300 most informative according to the preprocessing that was done by other researchers [51], and to change all digits and numbers to some corresponding text [46]. An instance of a test example

(100-300) often achieves effective results. We discuss some of these results later in the chapter.

of class *course* can be seen in Figure 5.2. LSI returns the correct classification of *course*, and

edu office rainbowthreedigit rainbowthreedigit rainbowthreedigit
rainbowthreedigit rainbowthreedigit rainbowtwodigit hours
rainbowonedigit rainbowonedigit rainbowonedigit
rainbowonedigit rainbowtime rainbowtime rainbowtime are are
be be be be course course course course home page page page
csrainbowthreedigit csrainbowthreedigit csrainbowthreedigit
csrainbowthreedigit algorithms spring this this this this for for for
and and and and and and and and and will will will of of of of of of
of data data structures material problems problems prerequisite
information handouts you you you you pm pm there is is is that
topics not lecture notes at exams final final related description
description rainbowfourdigit university computer computer
science science department introduction fall fall fall program
ming programming programming programming programming
programming code here or if if any newsgroup newsgroup
assignments development problem solutions solutions should
high credit about know me analysis rainbowtwodigit pm principal

Figure 5.2: Test example of class *course*

nearest neighbor returns the incorrect classification of *faculty*. To see why nearest neighbor returns the incorrect classification, we present the training example of class *course* in Figure 5.3, and the training example of class *faculty* in Figure 5.4. Counting simple overlap of words, the

office rainbowthreedigit rainbowthreedigit rainbowthreedigit
hours rainbowonedigit rainbowonedigit rainbowonedigit
note this for and of information homework homework
homework is is is class lecture notes exams grading description
rainbowfourdigit fall assignment welcome general syllabus
assignments project project announcements it solutions
know systems systems me current

Figure 5.3: Training example of class *course*

be course home home page page spring for for for and and and of of of of of design
room rainbowphonenum rainbowphonenum at at rainbowfourdigit rainbowfourdigit
rainbowfourdigit rainbowfourdigit university university computer computer computer
computer science edu edu rainbowthreedigit rainbowthreedigit rainbowthreedigit
rainbowthreedigit science science department department department programming
programming programming programming programming programming welcome
materials associate am am professor phone systems computing ph current my college
parallel analysis fax rainbowfivedigit director interests researchers ny finger advisor
advisor

Figure 5.4: Training example of class *faculty*

test example from Figure 5.2 shares 26 words with the training example of class *course*, but 40 words with the training example of class *faculty*. Even when TFIDF with normalization is used, the nearest neighbor algorithm returns a cosine similarity for the test example with the training example of class *faculty* of 0.502452 and a similarity with the training example of class *course* of 0.404355, and hence misclassifies this example. Using LSI, on the other hand, changes the space in such a way that the test example is closest to the training example of class *course*, with a cosine similarity of 0.797757 with the training example of class *course* and of 0.703648 with the training example of class *faculty*.

The example given above was simple in the sense that the training set consisted of only one example per class. However, this is usually not the case, and often multiple neighbors of the same class are returned by LSI. As in all nearest neighbor paradigms, the number k of nearest neighbors that are to be combined, as well as the combination rule itself are issues that must be dealt with. We approach these decisions in the same way as we did in Chapter 4 [14]. We can look at the result of the LSI query as a table containing the tuples

$$\langle \textit{train-example}, \textit{train-class}, \textit{cosine-distance} \rangle$$

with one line in the table per document in the training collection. There are many lines in the table with the same *train-class* value that must be combined to arrive at one score for each class. We once again use the noisy-or operation to combine the similarity values that are returned by LSI to arrive at one single value per class. If the cosine values for documents of a given class are $\{s_1, \ldots, s_n\}$, the final score for that class is $1 - \prod_{i=1}^{n}(1 - s_i)$. Whichever class has the highest score is returned as the answer to the classification task. Based upon [63, 14] only the thirty closest neighbors are kept and combined. This method of nearest neighbor in conjunction with LSI we term LSI-nn.

5.2 Incorporating Background Knowledge

5.2.1 The Expanded Space

The power of LSI lies in the fact that it can place documents that do not actually share any words in close proximity to each other. However, when there is little data LSI can suffer drastically. With few training examples, there are many terms that occur only once, hence limiting the power of LSI to create a space that reflects interesting properties of the data.

What is most interesting to us about the singular value decomposition transformation is that it does not deal with the classes of the training examples at all. This gives us an extremely flexible learner, for which the addition of background knowledge is quite easy. Instead of simply creating the term-by-document matrix from the training examples alone, we combine the training examples with other sources of knowledge to create a much larger term-by-“document” matrix, X_n. Figure 5.5 shows a schematic view of this new matrix.

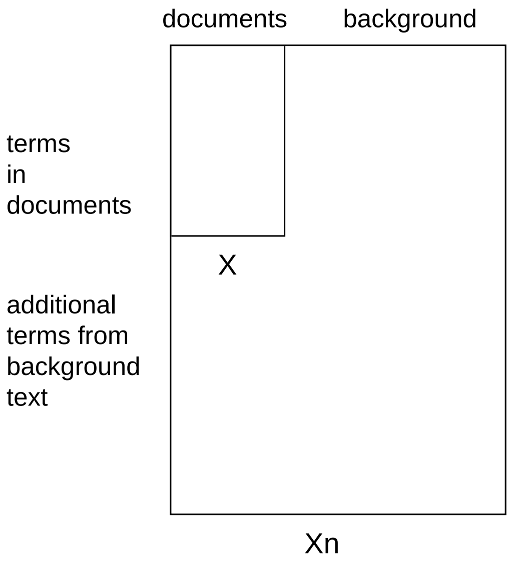


Figure 5.5: The expanded matrix incorporating background knowledge

Singular value decomposition is run on this new term-by-document matrix to obtain $\hat{X}_n$. $\hat{X}_n$ is a model of the space that was unobtainable with the training examples alone. The larger matrix contains words that did not occur in the training examples at all; it also provides us with richer and more reliable patterns for data in the given domain. To classify a test example while incorporating the background knowledge in the decision process, the test example is redescribed in the new space and then compared only to the columns of $\hat{X}_n$ that correspond to the original training examples. The scores that are obtained from this comparison are combined with the noisy-or operation, to return a final class for classification. We term this method of incorporating background knowledge into the LSI process LSI-bg. Clearly, it is important for the background knowledge to be similar in content to the original training set that it is combined with. If the background knowledge is totally unrelated to the training corpus, for example, LSI might successfully model the background knowledge, but the features would be unrelated to the actual classification task.

To give a concrete example of how LSI with background can help, we can look at one test example in the NetVet domain [14]. The training and test examples are titles of Web pages from http://netvet.wustl.edu, and each piece of background knowledge consists of the first 100 words of the contents of Web pages that are not in the training or test set. The training data in the example below consists of 277 documents. Removing all terms that occur only once creates a $t \times d$ matrix with 109 terms. With the added 1158 entries in the background knowledge the matrix grows to 4694×1435.

For the test example

british mule

of class *horse* the three closest training document returned were:

livestock nutrient manag univers

manag of the foal mare purdu univers

avitech exot

which are of class *cow*, *horse*, and *bird*. (In this example stemming [52] is used to find the morphological roots of the words in the documents for consistency of comparison with the WHIRL-bg approach). Since LSI creates a totally new space it is not unusual to find, as in this sample, that none of the original words from the test example are found in the three closest training examples. This test example is misclassified by LSI without background knowledge. This is not surprising since the word *mule* in the test example does not occur in the training examples at all. With the addition of the background knowledge, the three closest training examples returned are:

british columbia cattlemen

donkei

sicilian donkei preserv

of classes *cow*, *horse*, and *horse*. The correct class is returned. Notice that two of the closest training examples have the word donkei which is related to both *mule* and *horse*. The addition of the background knowledge allowed the learner to find this association.

5.2.2 Results of LSI-nn and LSI-bg

We obtained the Latent Semantic Indexing Package from Telcordia Technologies, Inc. (http://-lsi.research.telcordia.com/) and all results are with use of this LSI package. In Figures 5.6– 5.15 we report the classification accuracy for text classification using LSI both with and without background knowledge for all the data sets that were described in Chapter 2.

For all of the domains when the size of the training set is small, LSI-bg outperforms LSI-nn. For most of these domains the incorporation of background knowledge aided the classification task for training sets of *all* sizes. In most of the data sets it is true as well that the accuracy difference between LSI-nn and LSI-bg decreased as the training size increased. Also, although accuracy for both LSI-nn and LSI-bg decreased as the training set size decreased, the accuracy when using LSI-bg often changed very little as the size of the training set changed, as can be seen by the flatness of the lines representing LSI-bg (see for example Figure 5.14). This leads us to believe that the utility of background knowledge is that it compensates for the limited training data. This is true for all systems that use background knowledge, as has been pointed out by other researchers [51], but it can be particularly clear in the way that we use background knowledge in this chapter. This is because a limited amount of training data results in a $t \times d$ matrix of small dimensions, which does not allow the singular value decomposition process to find meaningful semantic associations. Both the addition of more training examples, or alternatively the addition of background knowledge, increases the size of the $t \times d$ matrix.

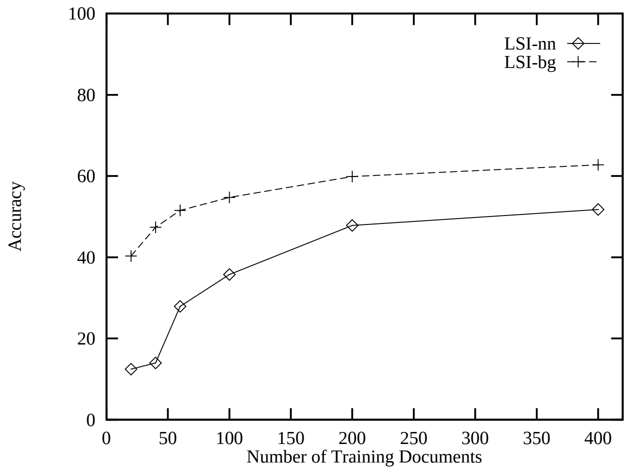


Figure 5.6: LSI-nn and LSI-bg for 20 Newsgroups

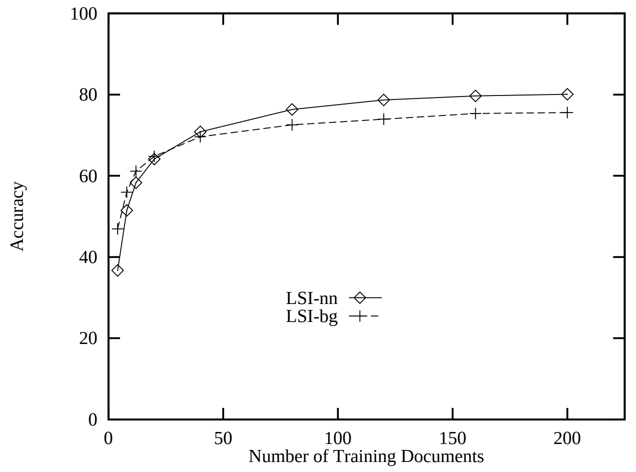


Figure 5.7: LSI-nn and LSI-bg for Webkb

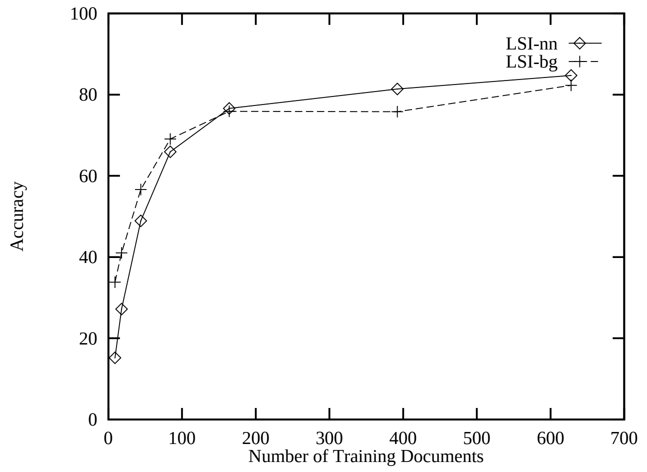


Figure 5.8: LSI-nn and LSI-bg for the advertisements problem

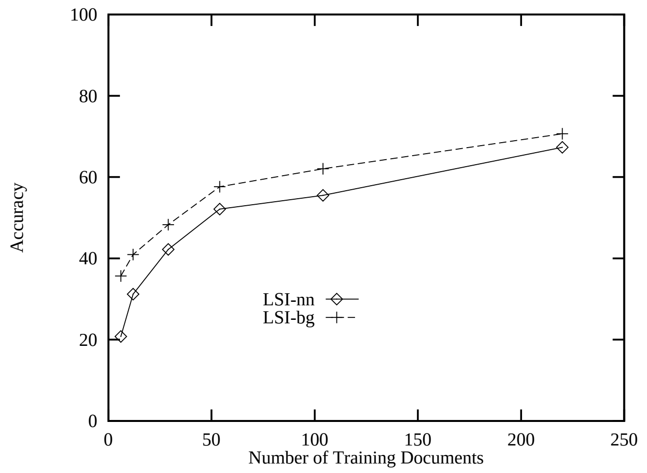


Figure 5.9: LSI-nn and LSI-bg for the ASRS problem

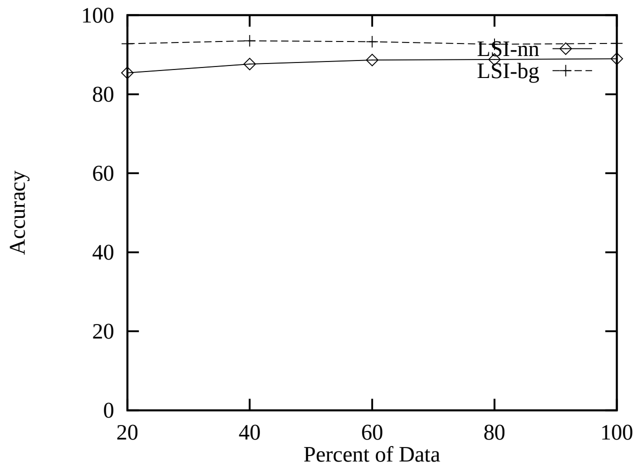


Figure 5.10: LSI-nn and LSI-bg for the 2-class physics title problem

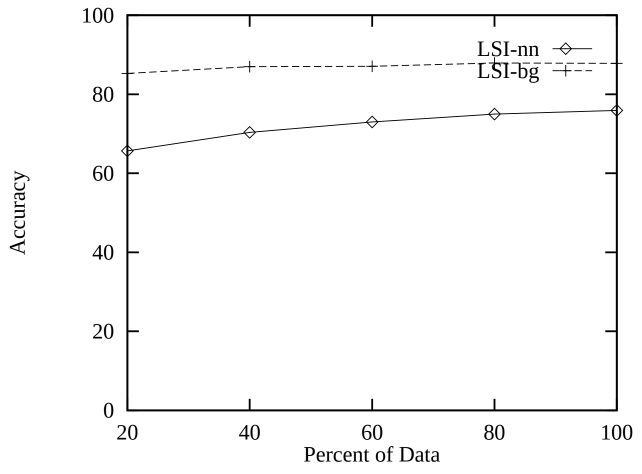


Figure 5.11: LSI-nn and LSI-bg for the 3-class physics title problem

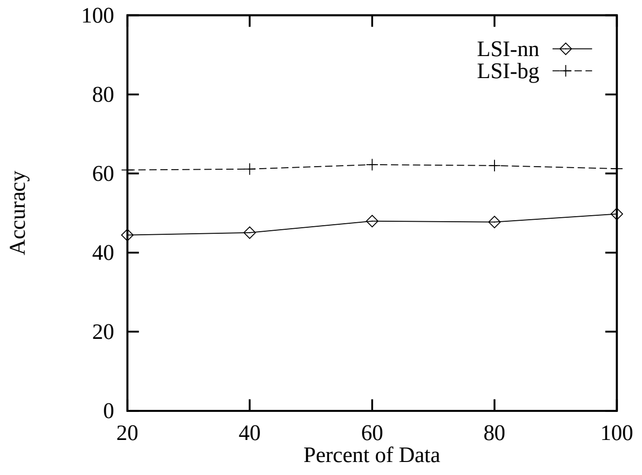


Figure 5.12: LSI-nn and LSI-bg for the NetVet data

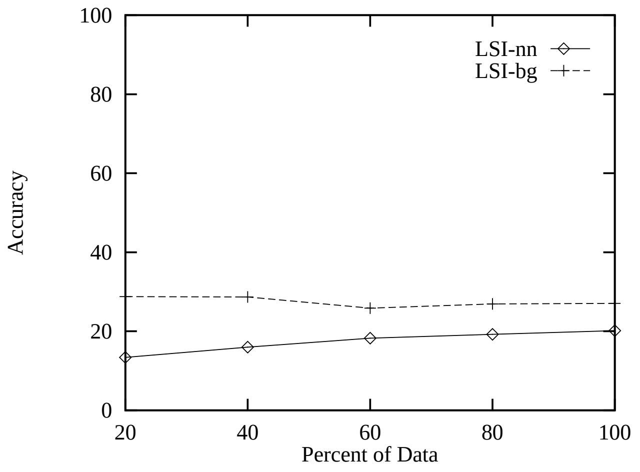


Figure 5.13: LSI-nn and LSI-bg for the business name data

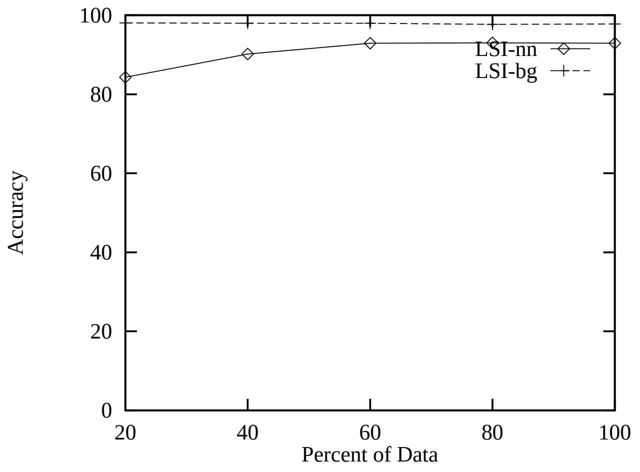


Figure 5.14: LSI-nn and LSI-bg for the Clarinet 9 problem

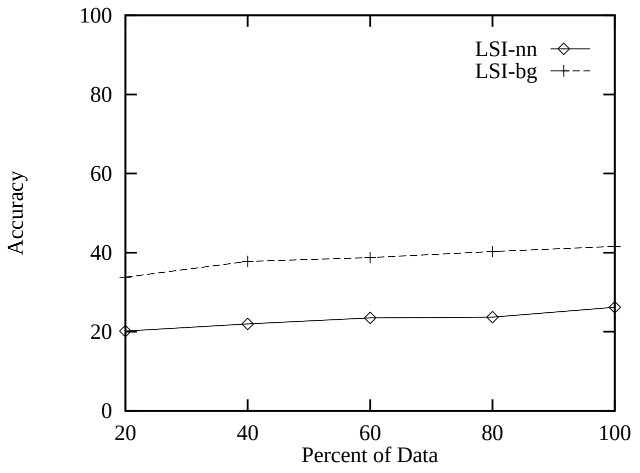


Figure 5.15: LSI-nn and LSI-bg for the thesaurus problem

Once again we wished to determine whether the difference in accuracy obtained using LSI vs using LSI-bg is statistically significant. From Figures 5.6–5.15 we can obtain 72 pairs of numbers. The first number in each pair represents the accuracy of running LSI while the second number is the accuracy of running LSI-bg on the same set of data. We used a paired t-test on these 72 pairs of numbers to determine if the addition of background knowledge caused a significant improvement in the accuracies, by checking the probability that the average difference between the pairs of numbers is consistent with zero. The resulting p value was less than .01, and hence we were able to conclude that significant improvements in accuracy are expected if background knowledge is added to the document matrix before singular value decomposition is done.

The choice of the number of factors for the examples that were presented above was 100. This choice of 100 factors was made based upon previous research in text classification [22]. It has been shown that a small number of factors (between 50 and 300) is useful in using LSI for test classification [22]. We ran LSI-bg using 50 factors, 100 factors, 200 factors and 300 factors on the thesaurus, NetVet, and 2 class physics titles data sets to ascertain the validity of our choice of 100 factors. Table 5.1 presents results for 20, 40, 60, 80, and 100 percent of the data for all of these datasets. We placed the highest value in bold face to show that it varies across the columns, however, as can be seen from this table, there is almost no difference in the accuracy rates across the rows, except for in the business name data. For the business name data, the smaller number of factors achieves higher accuracy than the larger numbers. This is probably the case because the vocabulary size (of words that occur more than once) of the training and test data is small, and the additional factors are probably simply modeling background information.

5.2.3 Incorporating the Test Examples

Since LSI is an unsupervised learner and it simply creates a model of the domain based upon the data that it is given without regard to the classes of that data, there are a number of alternative methods that we could use to enhance its power. One such method would be to include the test examples in the creation of the reduced space. If the original set of training documents is expanded to include both the training set and the test set, and SVD is run on this expanded matrix, hopefully more semantic associations would be found than if SVD was run on the training set

Table 5.1: Accuracy: Comparison of factor sizes with LSI-bg

Data Set	50 factors	100 factors	200 factors	300 factors
thesaurus-20	32.87	33.07	**34.17**	33.57
thesaurus-40	35.86	37.06	**40.26**	40.05
thesaurus-60	37.56	37.96	**40.56**	40.26
thesaurus-80	39.26	41.76	**41.55**	40.96
thesaurus-100	38.06	40.66	**42.95**	42.55
NetVet-20	**62.13**	61.53	60.86	59.77
NetVet-40	**62.09**	61.30	61.99	61.48
NetVet-60	61.13	61.46	**61.74**	61.59
NetVet-80	61.2	62.64	62.85	**63.03**
NetVet-100	62.28	62.97	63.16	**63.46**
Physics-20	92.44	92.55	92.13	**92.76**
Physics-40	**92.76**	92.66	92.02	92.76
Physics-60	92.03	92.24	**92.55**	92.23
Physics-80	92.34	92.55	92.86	**91.92**
Physics-100	92.65	92.45	**92.76**	92.76
Business-20	**29.89**	28.76	19.21	18.65
Business-40	**31.43**	28.68	20.55	20.79
Business-60	**31.03**	25.85	22.21	22.73
Business-80	**32.36**	26.94	23.87	23.75
Business-100	**31.88**	27.06	23.99	24.43

alone.

This type of learning is related to what is termed by Vapnik [61] as "transductive learning." Instead of simply using the training data, transduction makes use of the test examples in choosing the hypothesis of the learner. In the case of the nearest neighbor algorithm that we present in this chapter, we do not really find a *hypothesis* for the learner. However, the recreation of the space with the incorporation of the test examples does choose a *representation* based upon the test examples. This allows LSI to calculate entropy weights of words with the vocabulary and examples and co-occurrences of words in the test examples available.

We name LSI using the expanded matrix of the training examples plus the test examples, LSI-test. The singular value decomposition is done on this expanded matrix. To classify a test example using its representation in this new space, the test example is compared to all the training examples in this new space. The 30 training examples that are closest to the test example are then combined using the same noisy or operation that was described above. In Table 5.2 we

Table 5.2: Accuracy: Comparison of LSI-nn and LSI-test

Data Set	LSI-nn	LSI-test
thesaurus-20	20.18	22.68
thesaurus-40	21.98	25.77
thesaurus-60	23.47	27.27
thesaurus-80	23.6	26.77
thesaurus-100	26.17	28.47
NetVet-20	44.44	51.23
NetVet-40	45.04	53.75
NetVet-60	48.00	54.14
NetVet-80	47.73	54.96
NetVet-100	49.75	55.90
Physics-20	85.41	87.72
Physics-40	87.61	89.61
Physics-60	88.66	90.24
Physics-80	88.77	90.66
Physics-100	88.98	90.98
Business-20	13.39	19.38
Business-40	15.98	21.44
Business-60	18.29	22.69
Business-80	19.21	23.30
Business-100	20.14	23.91

present the accuracy rates for LSI-nn and LSI-test on three data sets. As we can see from the numbers presented, LSI-test always outperforms LSI-nn, although often by a small amount.

A major drawback of this method is that it assumes that the entire corpus of test examples is available when SVD is done, which is not always the case. The power of LSI-bg, as described above, is that related background knowledge is almost always readily available, or can be found quite easily from a variety of sources. In the examples that we present, for instance, the test sets are quite small in comparison with the size of the sets of background knowledge that we used. In the physics titles data set, the test set that was added for LSI-test consists only of 191 examples, while the background knowledge actually contains 1531 examples. In the NetVet set the test set size is 358 examples; the background contained 1158 instances. The thesaurus test set contained 200 examples, while the background knowledge contains 1007. The gains that could be made in accuracy by adding the test set into the singular value decomposition process is therefore limited, because the number of test examples that are available is limited. Also, in the case of training and test examples that are short text strings (as in the NetVet data set and thesaurus data set that we present) the number of terms that are added when the test examples are placed into the matrix is much smaller than when longer pieces of background knowledge are used. However, despite these deficiencies, it is still the case that we can look at the test set as an external corpus of background knowledge and use it alone, or in conjunction with other sources of knowledge to create the new semantic space. The test examples can be particularly useful in the LSI approach, because since the test examples are most closely related to the training set, we do not have to worry about modeling an unrelated concept that appears in the background knowledge, but that does not appear in the training and test set. The incorporation of the test set also allows the SVD process to place emphasis on terms that will be useful in the classification process, since they appear in the test set.

5.3 Summary

We have presented a method for incorporating background knowledge in text classification using LSI. The singular value decomposition is performed on a term-by-document matrix that includes both the training examples and background knowledge. This allows test examples to be compared to the training examples in a new space that reflects patterns in the text in the domain that may not be found when confronted solely with training data. We have shown empirically that this increases the accuracy rates in classification on range of problems.

Chapter 6

Comparing the Methods

In Chapters 3-5 we have presented three methods of incorporating background knowledge into the text classification task. Each of these methods uses the corpus of background knowledge in a different way, yet empirically, on a wide variety of text classification tasks we have shown that accuracy on test sets can be improved when incorporating background knowledge into these systems. However, it is the case that background knowledge can improve accuracy more in one method than another for any specific domain. In this chapter we look at how well each of these methods performed in comparison with the other methods. More importantly, we wish to point out how the choice of background knowledge affected each method's performance, and which type of background knowledge is most useful for each specific method.

6.1 When Does Each System Perform Well?

An important issue that we must address, aside from the misclassification *rates* of each of these systems on the various data sets, is an analysis of *which* examples are misclassified by the systems. Perhaps it is the case that each domain has a certain percentage of "hard" problems, which can never be classified correctly, regardless of the use of background knowledge or the type of learning system that is used. This is not always the case. We can look at some concrete numbers in the physics paper titles problem, which uses abstracts from physics papers as the background knowledge. Using our method described in Chapter 3 of five-fold cross validation, the five test sets consist of 190 examples each. Using 100% of the remaining data as the training set for each run, the average error rate for WHIRL-bg on this set is 5.7%. LSI-bg has an average error rate of 7.1%, and the EM method has an average error rate of 3.7%. If we look at the examples that are misclassified by EM, WHIRL-bg and LSI-bg, they are often not the same, even though the misclassification rates are very similar. If we look at the test examples that are misclassified by EM, which has the lowest error rate, an average of 55% of these misclassified examples are misclassified by WHIRL-bg as well. Of those misclassified by EM, only an average of 16.6% of these are misclassified by LSI-bg. Of the set that is misclassified by WHIRL-bg only 13.4% are misclassified by LSI-bg as well.

Interestingly enough, even within a specific system, the intersection of the set of examples that are misclassified with the inclusion of background knowledge and the set of examples that are misclassified without the use of background knowledge is often not large either. Once again, we can look at the statistics in the physics paper title example. Only an average of 25% of the test examples that were misclassified by WHIRL-bg are also misclassified by WHIRL-nn. For the other systems the intersection of the set of examples misclassified by the system with and without background knowledge is larger. 63% of the examples misclassified by LSI-bg were errors for LSI-nn as well. 65% of the examples misclassified by EM were misclassified by naive Bayes as well. Since EM uses the training examples to classify the background knowledge, in a sense the background knowledge is used through the view of the training examples. Test examples that cannot be correctly classified by EM most often cannot be correctly classified by the training examples alone either, since the training examples are both used in EM directly, and have been used to classify the unlabeled examples.

However, in Latent Semantic Indexing the use of background knowledge is completely different. LSI-nn creates a new space using only the training examples, and LSI-bg creates a new space using the combination of the training examples and the background knowledge. These two models of the data can be quite distinct and the reexpression of the training and test examples in these spaces can be very different as well. For that reason, we do not necessarily expect that the same test examples that are misclassified by LSI-bg are also errors for LSI-nn. However, when there is a large number of training examples, these spaces will be more similar and there will be more overlap in the error sets, as in the case of the physics paper titles using 100% of the training data.

For the system built on WHIRL, we do not expect the same set of examples to be misclassified by WHIRL-nn and WHIRL-bg. Since WHIRL-bg forces the use of background knowledge in the final decision for classification, WHIRL-bg relies more heavily than EM on the information contained in the background knowledge. Even if there are training examples that are close to a specific test example, and WHIRL-nn would classify the test example correctly, it is possible that it will be misclassified by WHIRL-bg because of the lack of background knowledge that is related to that specific test example. As an illustration of this possible occurrence, we can look at the Business name data. One of the 126 classes is *school*, and the test and training examples that are names of universities are classified as *school*. WHIRL-nn has no problem correctly classifying names of universities as class *school*, and this class has very low error. However, our background knowledge, which is taken from another business site, includes no universities (obviously the site that we obtained our background knowledge from does not consider universities as businesses!). This causes a problem for WHIRL-bg in its attempt to classify test examples of class *school*, and WHIRL-bg is forced to use background knowledge that does not really match well with these test examples. Hence, many of the test examples of this class are misclassified by WHIRL-bg.

These observations of the intersection of error sets leads us to believe that each of the systems have different strengths and weaknesses, allowing each system to classify problems correctly that other systems are unable to classify.

6.2 Comparisons of Results

The WHIRL-bg system performs best on the two problems where the form and size of the background knowledge is substantially different than the training and test data. In the business name data, the training and test data consist of short text strings that are names of businesses taken from the Hoovers (http://www.hoovers.com) site. Each piece of background knowledge, however, contains many different companies as well as descriptions of these companies, grouped together by Yahoo! business pages. These background pieces of data are not really classifiable, in the sense that they do not necessarily belong to any specific class in the Hoovers hierarchy. Since WHIRL-bg does not attempt to classify the background knowledge, but merely uses it to index into the training corpus, it makes the best use of this background knowledge. The same phenomenon is apparent in the advertisement data. The training and test data consist of individual advertisements taken from the Courier Post. In contrast, each piece of background knowledge consists of all advertisements under a specific topic from another web site (http://classifieds.dailyrecord.com). These two different sources do not use the same hierarchy, so not only are the background pieces of information of a different type, but they are not classifiable in the same way that the training and test data is. Once again WHIRL-bg outperforms the other systems in this domain.

For the data sets where the background knowledge fits very closely to the training and test classification task, EM outperforms the other systems. This is consistent with the way EM makes use of background knowledge. Since EM actually classifies the background knowledge, and uses the background knowledge to decide on the parameters of its generative model, the closer the background knowledge is to the training and test sets, the better EM will perform. Ideally, for EM, we wish the background knowledge to be generated from the same model as the training and test sets. The data sets that EM performs best on include 20 Newsgroups and WebKb, where the background knowledge is unlabeled examples, and hence of the exact same form as the training and test set. This group also includes the physics paper titles problems. Although the training and test data are titles are titles of papers, and the background consists of abstracts of papers, the background knowledge comes from the same site as the training and test sets, and is clearly classifiable in terms of which area of physics the associated paper belongs to. The NetVet data, to some degree, fits into this category as well. The training, test and background knowledge sets are all taken from the same web site, and each piece of background knowledge is associated with a page from a specific class in the problem.

LSI-bg seems to be most effective when there is limited training data. On the smallest data sets, LSI-bg outperforms all the other methods in many of the domains including 20 Newsgroups, NetVet, and two variations of Clarinet news (that will be introduced shortly), and is comparable to the best system for the smallest size training sets on some of the other data sets. We believe that this is so because LSI-bg is not dependent on the training data when creating its model. When very few training examples exist, LSI-bg can still build a space that correctly models the domain by using the available background knowledge. EM and WHIRL-bg depend more heavily on the small set of training data that is available; EM uses the training set to classify the background instances, and WHIRL compares elements of the training set directly to background instances. These two systems are therefore more affected by the limited training data. What is very interesting about LSI, is that with limited training data, LSI-nn often performs much worse than any other system. This is because the smaller space that LSI-nn creates can have very few factors and does not effectively model the domain. However, once the background knowledge is added, LSI-bg outperforms all other systems. For example, the error rate of naive Bayes on the 20 Newsgroups data with one training example per class is 20%, EM boosts it to 35%. In contrast, LSI-nn achieves only 12% accuracy, but LSI-bg has 40% accuracy. With the business name data, LSI-bg and WHIRL-bg both achieve 29% accuracy on the smallest data set, but LSI-nn has an accuracy of only 13%, while WHIRL-nn has an accuracy of 23%.

6.3 The Utility of Background Knowledge in Short Text Classification Tasks

The results that we presented in Chapters 3, 4, and 5 validate what many other researchers have found [51]: unlabeled data or background knowledge is most useful when there is little training data. This can be seen graphically in the figures presented in Chapters 3, 4 and 5 by looking at the distance between the lines that represent accuracy with and without background knowledge. In most of the graphs that we have shown, when the set of training data is small, the improvement upon the learning algorithm by including background knowledge is greater. As the number of examples in the training sets increase, the distance between the line that represents accuracy without background knowledge, and the line that represents accuracy on the test set with background knowledge decreases. A concrete example of why this is so can be seen from the Clarinet newsgroup dataset that was described in Chapter 2. Some test examples from this domain (stemmed [52]) are:

> instant replai fail again thi past weekend poor mike, sport. the career statist for the player involv in todai , sport.

When only 20% of the data is used for training, the first test example listed above would be misclassified to be of class *bank*. In this case, since there are only a few data points for training, the words *instant* and *replai* do not even occur in the training corpus at all. Since the word *poor* occurs in training examples from the *bank* category (as *Standard and Poor*) this example is misclassified. When background knowledge is used, and the words *instant* and *replai* occur often in the background knowledge in conjunction with other words from training examples in the *sport* category, this example is classified correctly.

There is another interesting point that has not yet been discussed, on the issue of when background knowledge could be most useful. As opposed to our discussion above, let us assume that we have a large number of training and test examples. However, co-occurrences of words may not be able to be learned properly and vocabulary size of the training set can still be small if each of the training and test examples themselves consist of very *short* strings. This might cause the same problems as an insufficient number of training examples would cause. We can intuitively understand that classification problems where the training and test data consist of short text strings will benefit most from the addition of background knowledge. As an example, we can look at the Business name data introduced in Chapter 2. Examples of training and test strings are:

abc inc, broadcast. watson pharmaceutical inc, drug.

It is clear from the type of examples in the training and test set that in this particular domain often the training data will be insufficient to classify a new test example. When data points in a text classification problem consist of only a few words each, it becomes hard for a learning system to obtain accurate counts of co-occurrences of words, as well as a complete vocabulary of words related to each class in the domain.

The Clarinet newsgroup problem described in Chapter 2 is a short-text classification problem that we created. Training, test and background knowledge sets are all taken from news articles in the two classes of *bank* and *sport* from Clarinet news. Although the training and test and background documents all come from the same source, we chose the training and test documents to be simply the first 9 words of the news articles. This reduces the size of the vocabulary that is in the training corpus, and also reduces the sharing of words between different examples. Since we chose each piece of background knowledge to consist of the first 100 words of news articles, each piece of background knowledge can overlap with many training or test examples as well as with each other.

To illustrate the effect that background knowledge has when the text classification problem consists of short-text strings, we have created three new problems from the Clarinet domain.

Table 6.1: Clarinet results on EM and Naive Bayes with different number of words in the training examples

Data Set	20%-Naive Bayes	20%-EM	100%-Naive Bayes	100%-EM
Clarinet-3words	77.4	89.0	85.2	91.3
Clarinet-9words	90.5	98.6	95.6	98.3

Instead of taking the first nine words from the news articles for the training and test data, we take the first 7 words, 5 words and 3 words to create the three new problems. We expect the problems with the shorter text strings to be helped more by the inclusion of background knowledge. A test example from the Clarinet news problem with 9 words, such as:

winner of saturdai tennesse kentucki football game still will, sport .

will be classified correctly by WHIRL-nn, making the background knowledge unnecessary for this particular test example. However, when only the first 3 words of the article are present the test example is:

winner of saturdai, sport.

and the words *football* and *games* are no longer part of the text. This causes WHIRL-nn to misclassify the problem, which can be compensated for by adding background knowledge, and the test example is properly classified by WHIRL-bg.

We plot the results for naive Bayes and EM, WHIRL-nn and WHIRL-bg, and LSI-nn and LSI-bg in the graphs where the problem names are called 3-words, 5-words, 7-words and 9-words, corresponding to the test and training set consisting of the first 3, 5, 7, or 9 words of each article respectively. Figures 6.1–6.4 graph the results of running naive Bayes and EM on these four problems. It is the case, as can be seen from the graphs, that EM improves upon naive Bayes for all of these problems, for any percentage of the training data that is used. As expected, naive Bayes performs worst in the 3-words problem, with 20% of the data, and best in the 9-words problem with 100% of the data. As can be seen from Table 6.1, the background knowledge raises the accuracy rate of the 3-words problem with 20% of the data from 77.4% to 89% and the 9-words problem with 20% of the data from 90.5% to 98.6%. With 100% of the data, the background knowledge raises the accuracy of the 3-words problem from 85.2% to

91.3% and the 9-words problem by only 2 percentage points, from 96.6% to 98.3%. We can see that both the addition of training examples, as well as the addition of more words per each training example, limits the usefulness of the background knowledge.

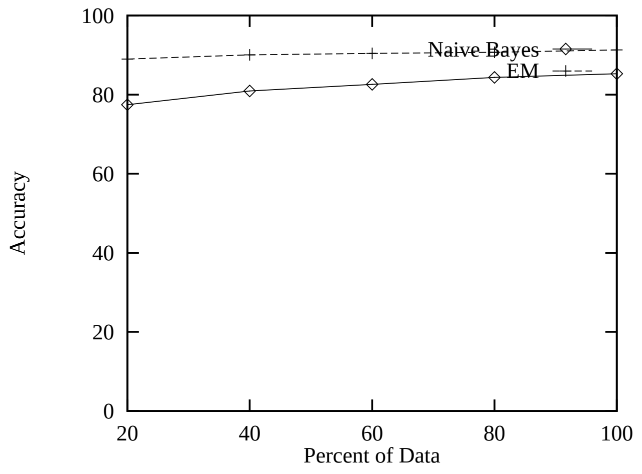


Figure 6.1: Naive Bayes and EM for the Clarinet 3 words problem

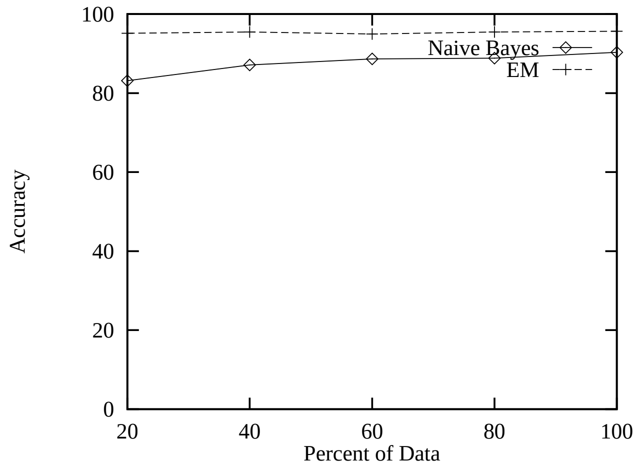


Figure 6.2: Naive Bayes and EM for the Clarinet 5 words problem

Results on the Clarinet data set for WHIRL-nn and WHIRL-bg are presented in Figures 6.5–6.8. As expected, the smaller the number of words in each training and test example, the worse both WHIRL-nn and WHIRL-bg performed. The addition of background knowledge was most useful with the shorter strings in the test and training data as well. This is represented in Figures 6.5–6.8 by the point at which the two lines intersect. For strings of length 3, background knowledge reduced the error rates, even when the entire set of training data was used. As the

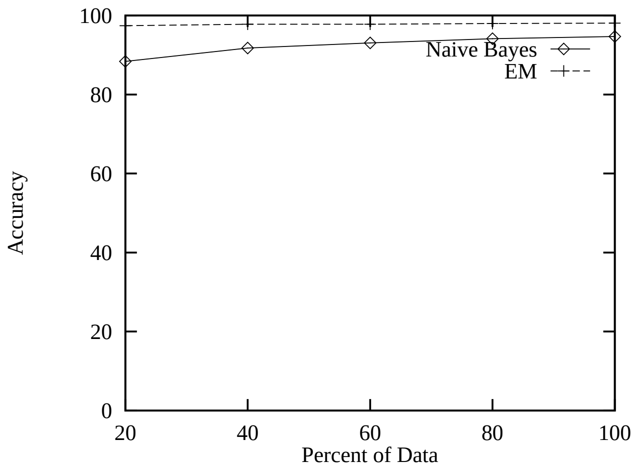


Figure 6.3: Naive Bayes and EM for the Clarinet 7 words problem

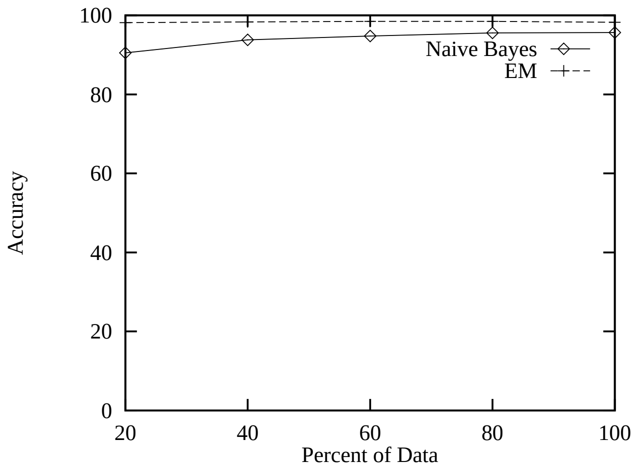


Figure 6.4: Naive Bayes and EM for the Clarinet 9 words problem

number of words in the training-test examples increased, the point at which background knowledge became helpful changed. For strings of length 9, background knowledge reduced error rates only when less than 60 percent of the data was used. This gives empirical evidence that the less informative the training data is, the greater the advantage in having a corpus of background knowledge available for use during classification. The size of the reduction in error rate obtained by running WHIRL-bg was also greater when there were fewer words in each example.

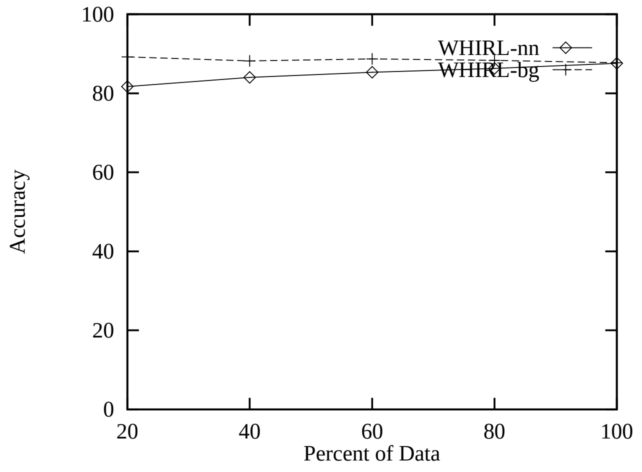


Figure 6.5: WHIRL-nn and WHIRL-bg for the Clarinet 3 words problem

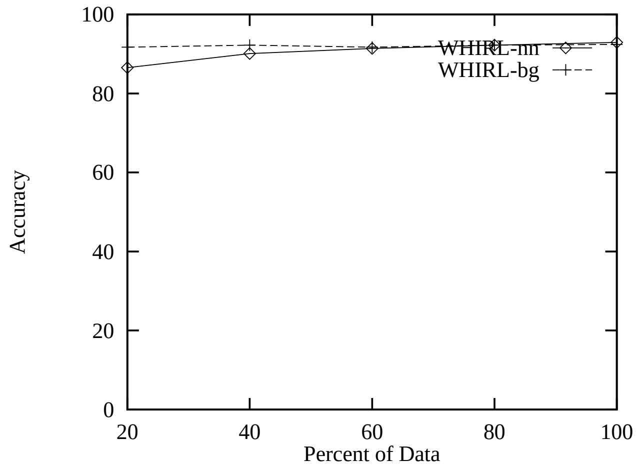


Figure 6.6: WHIRL-nn and WHIRL-bg for the Clarinet 5 words problem

Results for LSI-nn and LSI-bg on the four Clarinet problems are graphed in Figures 6.9–6.12. On all four of the data sets the difference in the line representing LSI-nn and LSI-bg is larger when the data set is smaller. The unlabeled data is more useful when there are fewer training examples. However, it is also the case that the difference in the lines is greater for the

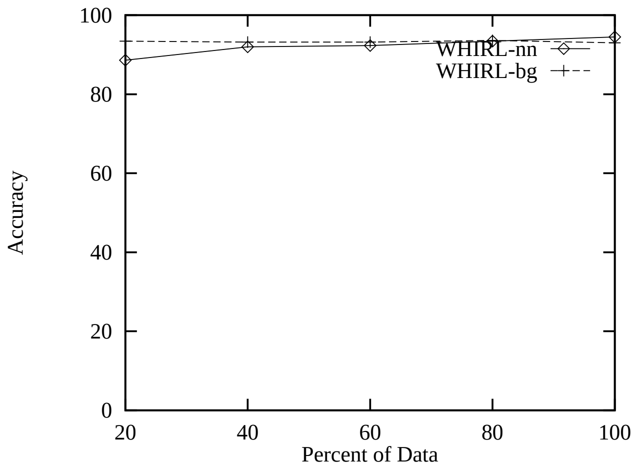


Figure 6.7: WHIRL-nn and WHIRL-bg for the Clarinet 7 words problem

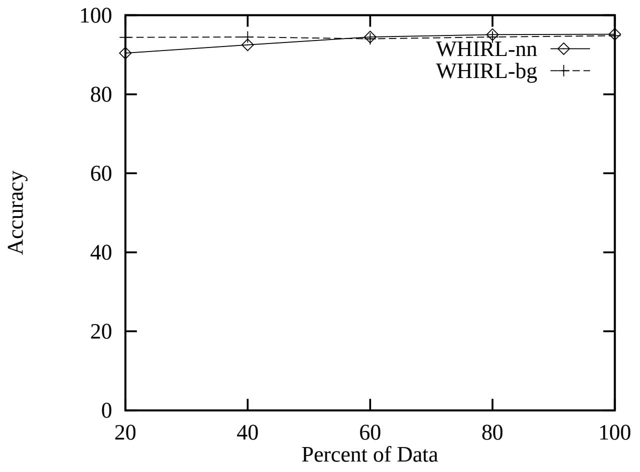


Figure 6.8: WHIRL-nn and WHIRL-bg for the Clarinet 9 words problem

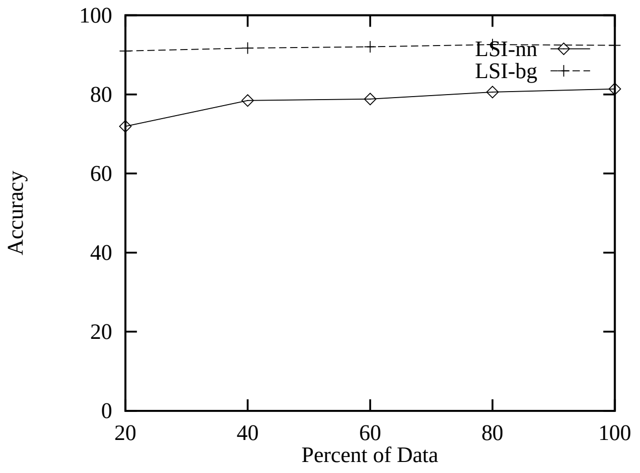


Figure 6.9: LSI-nn and LSI-bg for the Clarinet 3 words problem

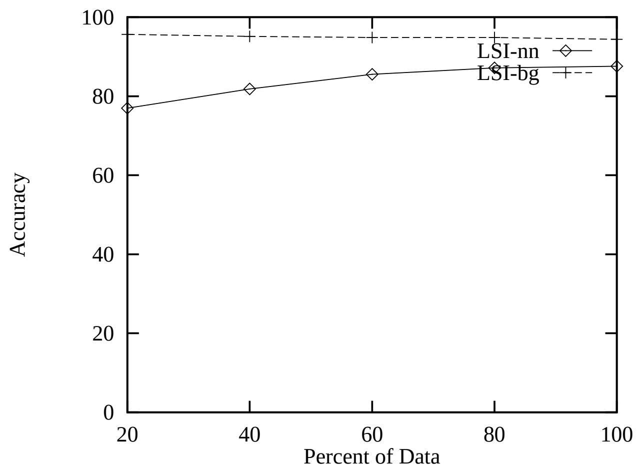


Figure 6.10: LSI-nn and LSI-bg for the Clarinet 5 words problem

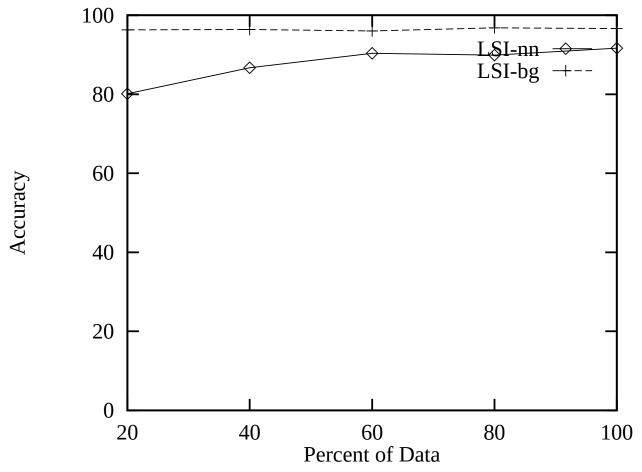


Figure 6.11: LSI-nn and LSI-bg for the Clarinet 7 words problem

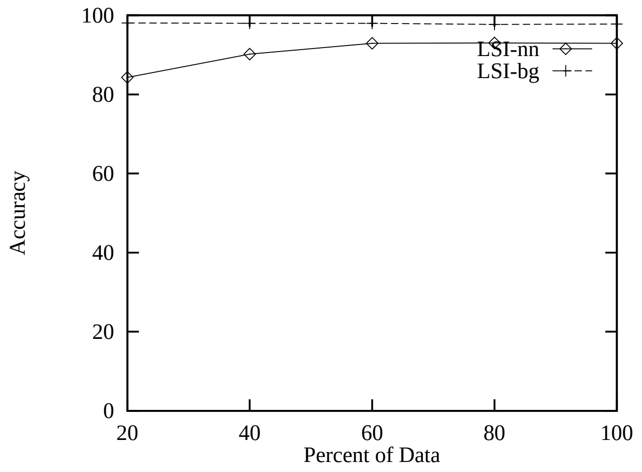


Figure 6.12: LSI-nn and LSI-bg for the Clarinet 9 words problem

shorter text problems. With 100% of the data on the three word problem, LSI-bg has an error rate of 90%, and LSI-nn an error rate of 71%. This difference is 19%. For the five word problem, the error rates are 77% and 96% for a difference of 19% as well. However, the 7 words problem has error rates of 80% and 96% for a difference of 16% and the 9 words problem has error rates of 84% and 98% for a difference of only 14%. The same phenomenon is noticeable on all training set sizes, with the usefulness of background knowledge being more apparent in the shorter text string problems as the training set size decreases. For 20% of the data, LSI-bg boosts the accuracy rate by 11%, 6%, 5% and 5% for the 3-words, 5-words, 7-words and 9-words problems respectively. This is because a training data of short text string results in a $t \times d$ matrix of small dimensions because of the smaller vocabulary size, which does not allow the singular value decomposition process to find meaningful semantic associations. The expansion of the training data to longer text strings, the addition of more training examples, or the addition of background knowledge (or any combination of these three), increases the size of the $t \times d$ matrix.

6.4 Unrelated Background Knowledge

In Chapter 4 we discussed the effect that unrelated background knowledge has on WHIRL-bg. Since WHIRL-bg compares a training and test example only through a piece of background knowledge, if the background knowledge is unrelated to the task, the accuracy of WHIRL-bg will often be lower on the test set than even WHIRL-nn. We introduced the idea of a disjunctive query, which we termed WHIRL-dis, that can be used instead of WHIRL-bg if the background knowledge is not known to be from the same domain as the training and test sets. WHIRL-dis includes a comparison between the training and test sets directly, and so minimizes the negative effect that unrelated background knowledge can cause. In this section, we present experiments to see how EM and LSI deal with unrelated or mixed background knowledge.

We present graphs on the four data sets that we used to test WHIRL-dis: the thesaurus problem, the business names problem, the 2-class physics problem and the NetVet problem. We ran each of these data sets without background knowledge, with the correct related set of background knowledge (as described in Chapter 2), with a mixed set of background knowledge that contained both the correct background knowledge and additional unrelated background knowledge and with only the unrelated background knowledge. For the unrelated background knowledge we use the background set from the NetVet data for the other three tasks, and the physics abstracts for the NetVet task. The mixed background set consists of all documents in the related background set plus all documents in the unrelated set of background knowledge for each task.

Figures 6.13– 6.16 present the results of running WHIRL with different sets of background knowledge. Each graph contains five lines, one for WHIRL-nn with no background knowledge, WHIRL-bg with the correct set of background knowledge, WHIRL-bg with the mixed set of background knowledge, WHIRL-bg with the wrong set of background knowledge, and for comparison sake, WHIRL-dis with the wrong set of background knowledge. As can be seen from the accuracy curves, WHIRL-bg with the correct set of background knowledge outperforms all other runs, as is expected. WHIRL-bg with unrelated background knowledge performs worse than any other method, including WHIRL-nn. However, the curve for WHIRL-dis with the unrelated background knowledge is very close (and even overlapping sometimes) to the curve for WHIRL-nn. This revalidates our claim that WHIRL-dis reduces the degradation of accuracy that unrelated background knowledge can cause with WHIRL-bg.

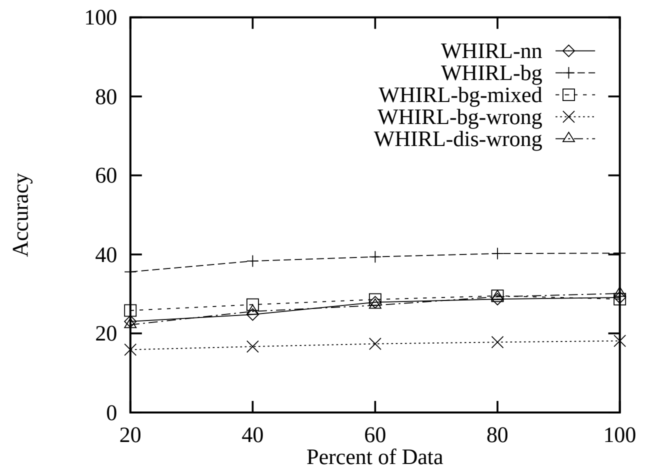


Figure 6.13: Using WHIRL with different sets of background knowledge on the business names data

Figures 6.17– 6.20 present the results of running Naive Bayes and EM. Each graph consists of four curves: one for Naive Bayes, one for EM with the correct set of background knowledge, one for EM with the mixed set of background knowledge, and one for EM with the wrong set

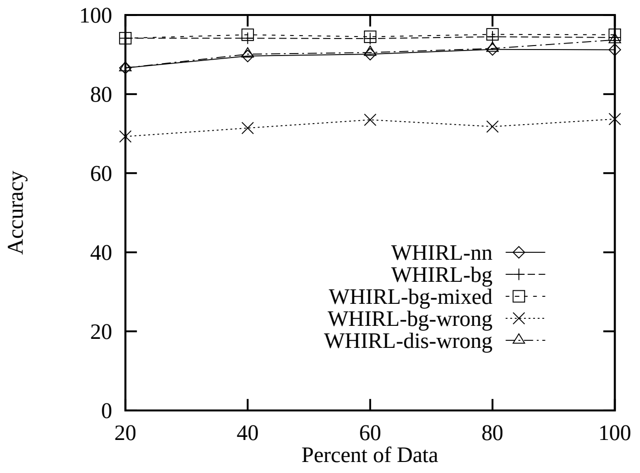


Figure 6.14: Using WHIRL with different sets of background knowledge on the physics 2-class data

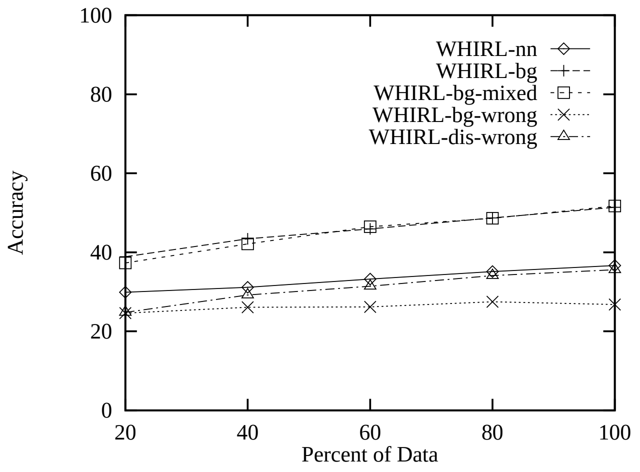


Figure 6.15: Using WHIRL with different sets of background knowledge on the thesaurus data

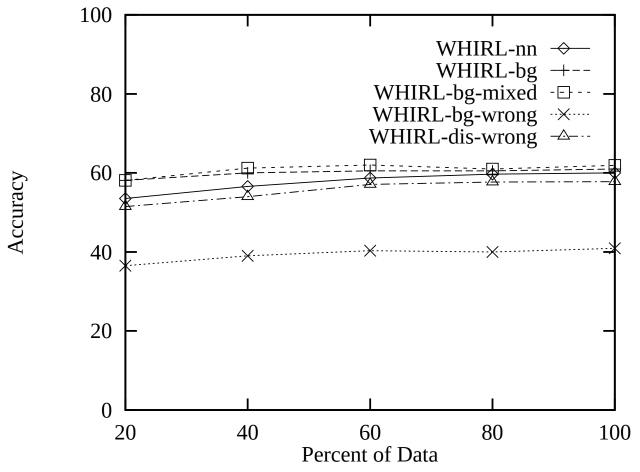


Figure 6.16: Using WHIRL with different sets of background knowledge on the NetVet data

of background knowledge. What is interesting is that in all four cases, the mixed set of background knowledge does not cause accuracy to be worse than Naive Bayes. In the physics data and thesaurus data, EM with mixed background performs as well as EM with the correct set of background knowledge. Even with the wrong set of background knowledge, EM does not perform more poorly than Naive Bayes on the business names and physics data. If the iterations in EM do not classify the background knowledge as belonging with high probability to any class, it will minimize the effects that this background knowledge will have on the final model parameters. In the NetVet and thesaurus data sets, EM with the wrong background knowledge does perform worse than Naive Bayes. However, our version of EM is the straight forward and simple one. Nigam et al. [51] present two extensions to EM that might minimize the effect of wrong background knowledge. Specifically, if the weights of the unlabeled examples in terms of their contribution to the model parameters is reduced, misleading background knowledge would probably have less of an effect on accuracy.

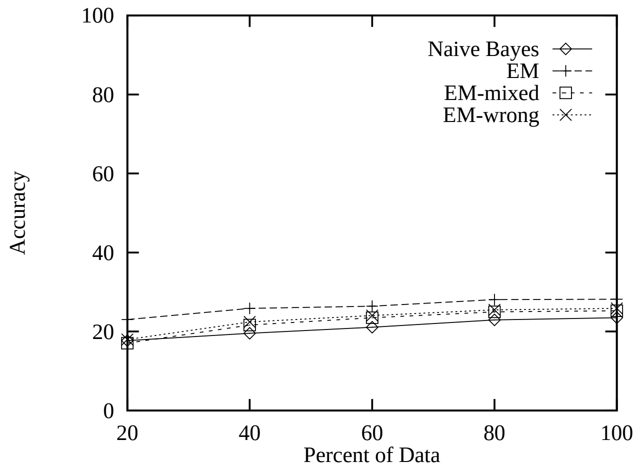


Figure 6.17: Using EM with different sets of background knowledge on the business names data

We present the results of running LSI with different sets of background knowledge in Figures 6.21–6.24. Once again, each graph consists of four curves, LSI-nn, LSI-bg with the correct set of background knowledge, LSI-bg with the mixed set of background knowledge and LSI-bg with the wrong set of background knowledge. As can be seen from the graphs, the use of unrelated background knowledge can cause extreme results with the LSI approach. This was not surprising to us, since LSI uses the training set in combination with the background knowledge

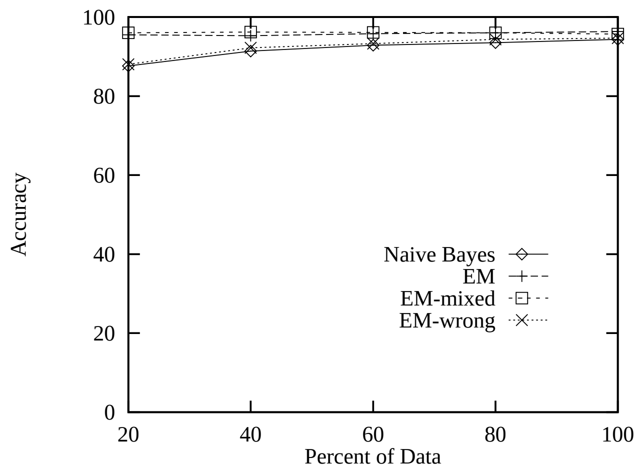


Figure 6.18: Using EM with different sets of background knowledge on the physics 2-class data

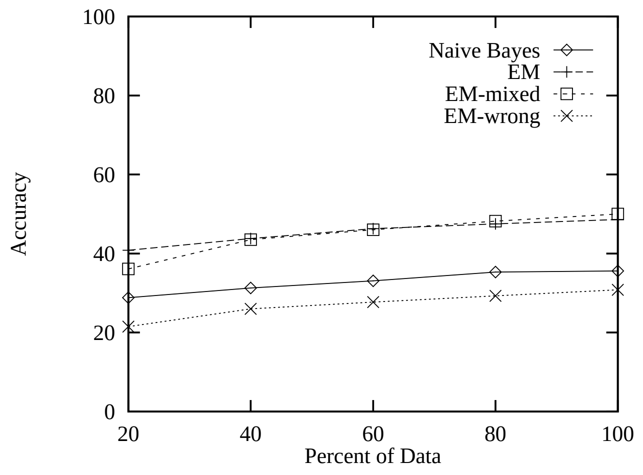


Figure 6.19: Using EM with different sets of background knowledge on the thesaurus data

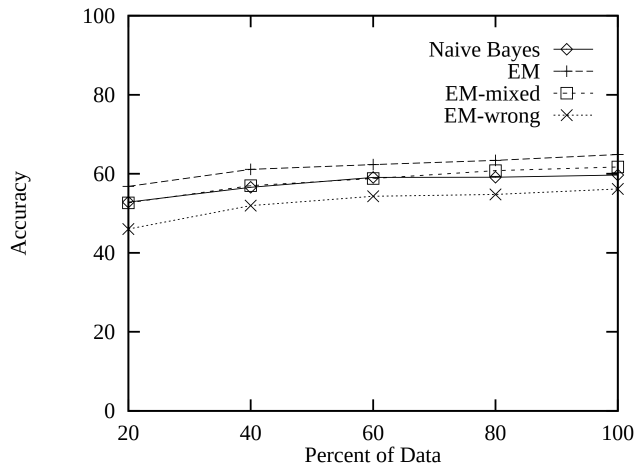


Figure 6.20: Using EM with different sets of background knowledge on the NetVet data

to create a new space. If the new space reflects the training set, or correct set of background knowledge, then the comparisons between the training and test examples can provide accurate results. However, if the new space reflects the unrelated background knowledge, and the training and test examples are re-expressed in this unrelated space, comparisons between the training and test examples can be meaningless.

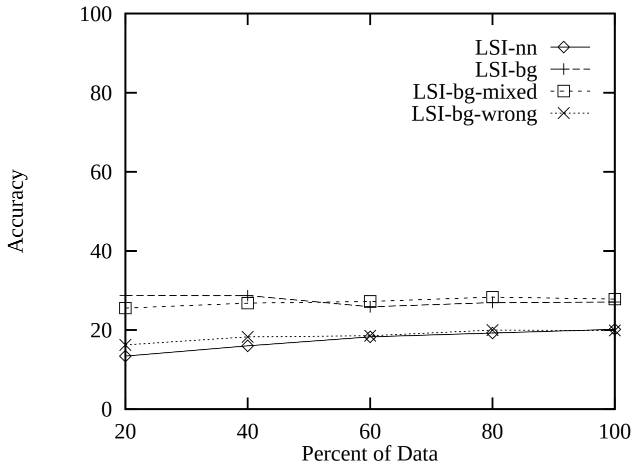


Figure 6.21: Using LSI with different sets of background knowledge on the business names data

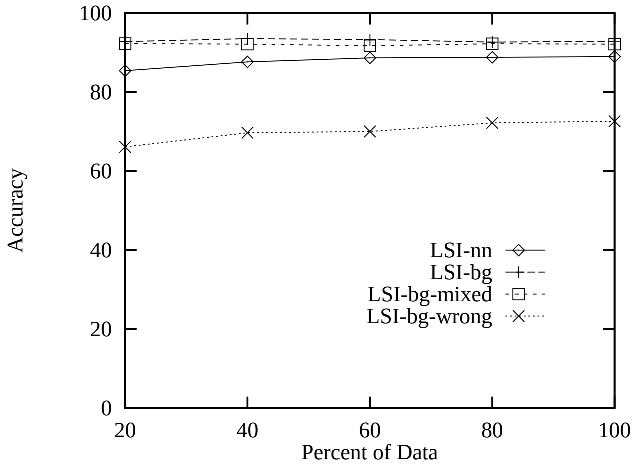


Figure 6.22: Using LSI with different sets of background knowledge on the physics 2-class data

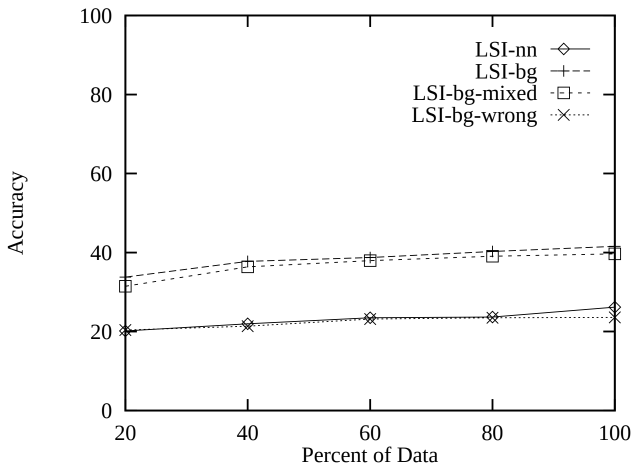


Figure 6.23: Using LSI with different sets of background knowledge on the thesaurus data

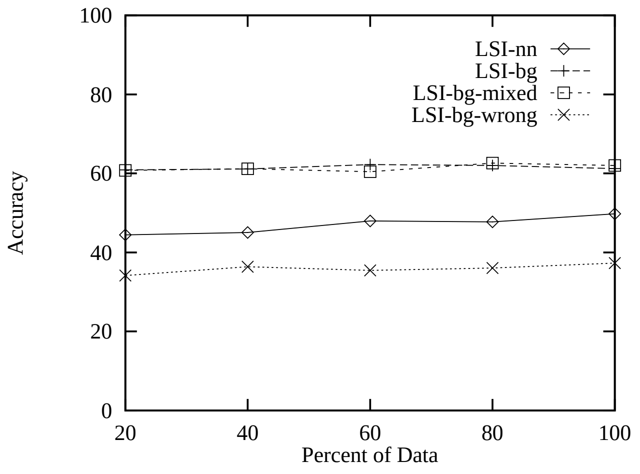


Figure 6.24: Using LSI with different sets of background knowledge on the NetVet data

6.5 Summary

In this section we analyzed when and where each method that we presented in the previous chapters performed best. We showed that often examples that are misclassified by one method could be properly classified by another method. We presented results for all three methods on different lengths of training and test data, to show that background knowledge helps most when the classification task is one that deals with short text strings. Finally, we discussed how irrelevant background knowledge affects the performance of each method.

Chapter 7

Related Work

Throughout our thesis we have discussed numerous pieces of related work that was directly necessary for the explanation and discussion of our innovative ideas. Chapter 3 gives a discussion of naive Bayes classifiers, Chapter 4 discusses WHIRL and other text classification tools that are used for comparisons, and Chapter 5 includes descriptions of related work in Latent Semantic Indexing. However, there is other work that in the area of combining supervised and unsupervised learning that must be mentioned.

Although the concept of background knowledge that we presented in this thesis is novel, the ongoing work on the addition of unlabeled examples in supervised learning algorithms is very closely related to our research. As a matter of fact, as we pointed out in Chapter 3, unlabeled examples can be looked upon as a limited type of background knowledge in the context of our experiments. Most of the work on unlabeled examples has been empirical in nature, and it can be shown that unlabeled examples can improve classification accuracy when added into many different classifiers [51, 33, 31, 64, 65, 7, 6, 48]. Theoretically the usefulness of unlabeled examples is often unprovable even when dealing with unlabeled examples that are extremely "similar" to the labeled data. Since our methods make use of background knowledge that is often qualitatively different than the training and test set there are no theoretical results that can be straight-forwardly applied to determine how useful this knowledge would be in classification.

Transduction, as introduced by Vapnik [60, 61], makes use of the test examples in choosing the hypothesis by the learner. Joachims [33] presents an algorithm for transductive SVM that chooses, from among all the possible hyper-surfaces, the one that correctly classifies the data while maximizing the margin[1] of both the training and test data. In a sense the use of the test data during the text classification task injects some prior knowledge of words that can be used

[1]The margin is defined to be the distance between the hyper-surface and the nearest examples of the given classes.

in the decision of the choice of the hyperplane that would be used for classification. Joachims presents results for text classification tasks that show the validity of this approach. Results by others [2] make use of what they term a “working set”, which is essentially a corpus of unlabeled data, in SVMs as well. Their empirical results on ten data sets showed that transductive SVM via overall risk minimization (large margins on both the training and test set, as well as accuracy on the training set) improved induction over structural risk minimization (large margins on the training set, as well as accuracy on the training set) some of the time and never degraded the learner. However, later results by others [66] question the efficacy of this approach. Wu et al. [62] apply transduction to perceptron decision trees. Each node in these perceptron decision trees represents a hyperplane in the input space. These trees often suffer from overfitting and to combat this problem often the trees are pruned to be made simpler. Wu et al. [62] show that by choosing the decision nodes that both maximize the training accuracy and take into account the margins on the training data, higher accuracy on the test data can be achieved. Although they show that this margin maximization is useful in the inductive case for pruning and to reduce overfitting, the benefits of transduction (i.e. the inclusion of the test set into the choice of the hyperplanes) are less clear.

A different approach to combining labeled and unlabeled information is taken by Blum and Chawla [5]. They create a graph from the examples by connecting all labeled examples to the node corresponding to the class assignment and to each other based upon their similarities. Unlabeled examples are connected to their nearest neighbors. The min-cut algorithm is then used to find the best division of the data into positive and negative classes. Issues of how to measure similarities and how many edges to connect to each unlabeled example are some of the directions of current research.

The co-training paradigm (Blum and Mitchell) [6] takes advantage of the fact that a data set can often be expressed in more than one way. For example, a web page can be described by the words on the page, or by words in hyper-links that point to that page. This structural knowledge about the data can be exploited to use unlabeled data. Initially, two hypothesis, H_1 and H_2 are formed on each of the views using the labeled data. A subset of the unlabeled data is then classified alternatively by h_1 and h_2 and added to the other view to be part of the labeled set. Assuming that the two views of the data are conditionally independent of each other, Blum and Mitchell [6] prove that a weak initial predictor can be used with unlabeled data to PAC learn [59] the target concepts. Nigam and Ghani [50] extend work on co-training and show that when independent and redundant expressions of the data are available, co-training improves on other algorithms. Even when no such expressions of the data are available, co-training is robust in that it can improve learning when the features of the data are divided randomly between two learners.

The co-boosting algorithm, an extension of the AdaBoost supervised classification algorithm, is presented by Collins and Singer [15] for the task of named entity representation. Once again, the examples are expressed with different views. Goldman and Zhou [28] dispense with the notion that two independent views must be created from the data, and instead use different learners on the same view. They argue that although the data is the same for each of the algorithms, by virtue of their very different natures, the two supervised learning algorithms can complement each other and be combined in a useful way.

Unlabeled data has been used in text classification task where no labeled data is available, but a class hierarchy or set of keywords per class is known [44, 54]. In these cases, the keywords that are given for each class are used to assign preliminary labels to the documents. These labels are then used as a starting point for expectation maximization and naive Bayes in conjunction with unlabeled examples to arrive at a set of parameters for a generative classifier.

Cohen [13] uses a concept that is somewhat similar to our idea of background knowledge. Cohen [13] uses Web pages in HTML to create new features for training and test examples in a machine learning task. A group of Web pages that is related to the training and test data is used to create tuples based upon the HTML headers or positions in the HTML document and values under the header. New features are then are added to training and test examples if the values of data in the examples are similar to values in the tuples. These Web pages can be looked at as background knowledge that is used to aid the learning task.

Chapter 8

Future Work and Conclusions

8.1 Evaluation of Background Knowledge

The nature and type of background knowledge that is used to improve learning is of central interest to us. The data sets that we used in our experiments had background knowledge of different kinds, from a variety of sources. There are a number of issues that we can look at in our evaluation of different types of background knowledge.

- Are unlabeled examples more helpful than background knowledge that comes from a different source? For unlabeled examples the size of each piece of background knowledge is generally well-defined since each piece of background knowledge is simply of the same general size and type as the training and test examples. We can also be more assured of the relationship between this type of unlabeled examples and the text classification task. However, there is a disadvantage in the use of unlabeled examples when dealing with short-text classification tasks. When the training and unlabeled set consist of examples that have only a few words, a set of unlabeled examples does not allow us to learn as many co-occurrences and frequencies of words as a set of longer pieces of background knowedge would. This limits the usefulness of unlabeled examples as background knowledge.

  The use of unlabeled examples as background knowledge also assumes the easy availability of unlabeled examples. Although it is often the case that unlabeled examples are plentiful, it is not always so. When unlabeled examples are not available, our only choice may be to use background knowledge from a different source.

- The choice of a source for the background knowledge is crucial as well. There has been some work done by Dumais et al. [24] in cross-language retrieval using Latent Semantic Indexing. The work uses a corpus of textual documents, where each document consists of the text in both English and French. The semantic space represents words in both languages and queries in either language can be represented and answered in either French or English. Dumais et al. [24] give results of experiments using three different corpora of translated documents. When the corpus of documents was different than the queries (using parliamentary papers as the corpus) the system performed worse than when the corpus was more related to the queries (using the yellow pages as the corpus). Ideally we would like to be assured that the vocabulary of the background knowledge overlaps in a great measure with the training and test sets. This is most crucial for the LSI method, as well as for the naive Bayes, but would be important for WHIRL as well.

The question that must be posed is: "How close is the background knowledge to the task?" WHIRL (or any other k-NN algorithm) will allow us to determine the closeness of background knowledge to the training and test set. Our WHIRL-bg algorithm actually incorporates this metric into its second-order approach. A question for future work that we wish to explore is if this metric can be used to determine the closeness of the background knowledge to the task for other methods as well. Perhaps it can it be used in some preprocessing algorithms to determine if the background knowledge will be useful, or at least to figure out if background knowledge will not cause accuracy on the test set to degrade. These are issues that we are currently exploring. The "cleanliness" of background text can also vary greatly, from encyclopedia entries at one end of the spectrum to ad hoc collections obtained from uncoordinated Web sites or text obtained through speech recognition technology on the other. We would expect the cleanliness of the data to impact the usefulness of the background knowledge as well.

- The granularity, or amount of information that each individual piece of background knowledge contains is also an important area to examine. In our advertisement data set and Business names data set the pieces of background knowledge were of a very different size than the training and test data. In each background piece of data contained many entries that were similar to a training or test example. This type of background knowledge is easier to obtain and download from the Web, as they do not have to be parsed as carefully as other types of background knowledge that mimic the form of the training and test data. The granularity of the background knowledge also may help determine which type of learning algorithm might be most useful. For instance, as the form of the background knowledge deviates more from the form of the training and test data, the EM algorithm might be less useful, as it attempts to treat the background knowledge as actual examples.

- Another question concerns the combination of different sets of background knowledge. In the simplest approach, if we obtain two sets of background knowledge (or two background knowledge databases), we can simply concatenate the two sets and use it as one large body of background knowledge. If both sets are related to the task, this is simply an issue of enlarging the size of the background knowledge data base. However, suppose we are more certain about one background knowledge set in the sense that it contains more reliable information. For example, for the physics paper title problem, abstracts from technical papers might contain information and words that are more reliable than a newsgroup discussion about issues in physics. We would therefore like to combine the two sets of background knowledge by weighting one more highly than the other. Issues in this area that we are exploring include how to add the weighting to the systems, and how to explore a correct weight for each set of background knowledge.

- A key issue that we have not addressed in this work is *how* we obtain the background knowledge. In this thesis, we have manually chosen background knowledge for each learning task that we explored. We have done some preliminary work on automatically generating this background knowledge via search utilities on the Web. One simple method that we have looked at uses some important key words from the domain (e.g. in the NetVet domain, these might be the names of the animals that are also class names), and does a Web search on each of those words. Pages returned from the searches are used as entries in the background knowledge database. The problem with this approach is that the background knowledge comes from many different sources, with a wide vocabulary range. This makes it harder for our systems to find patterns or co-occurrences, and our initial empirical results on this are mixed.

8.2 Extensions and Modifications

8.2.1 WHIRL-bg

It is interesting to note that our use of background knowledge in a WHIRL query is in a sense a form of query expansion [8]. Instead of directly searching for the training examples that are closest to a test example we search for the training examples that are closest to the background knowledge expansion of the test example. However, unlike standard query expansion, and because of our conjunctive conditions, the background knowledge expansion itself is chosen with respect to the training example that it is close to. This means that each query has multiple expansions, and all those that maximize the score of the conjunctive condition are combined.

Given a query (or test example in our discussion) we can say that we expand the query by substituting it with a piece of background knowledge that it is close to. This expansion is only used if there also exists one or more training examples that is close to this expansion. We expand the query in many different ways, substituting any background piece of knowledge for the query that has a training example that is close to it as well. We leave the exploration of the use of our WHIRL-bg approach for retrieval, as opposed to classification, for future work.

We will also begin to look at further extensions of our approach. For example, consider the query:

```
SELECT Test.instance, Train.label
  FROM Train AND Test AND Background as B1
    AND Background as B2
WHERE Train.instance SIM B1.value
    AND Test.instance SIM B2.value
    AND B1.value SIM B2.value
```

This type of query provides a different way for background knowledge to bridge gaps between a training example and the test example. Given that the test and training examples only have a small bit of knowledge about the class to which they belong, this query allows each small bit of knowledge to be mapped to larger pieces of background knowledge that can then be compared to each other. Our current version of WHIRL-bg connects a training example and a test example by using a bridge through a piece of background knowledge. This bridge can only be formed if the it same piece of background knowledge is close to both the training and test example. If the vocabulary of the background knowledge is much larger and more varied than that of the training and test set, it is possible that *one* piece of background knowledge will not be close to both a training example and the test example. However, it might still be the case that the piece of background knowledge that is close to the training example and the piece of background knowledge that is close to the test example share many words with each other. In a sense, to use the bridge analogy, this query creates a bridge with two steps by comparing the background pieces of knowledge with each other (Figure 8.1).

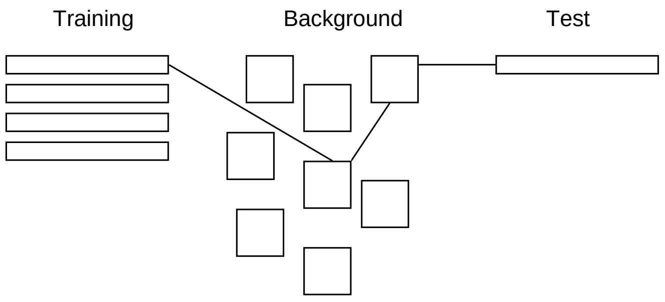


Figure 8.1: WHIRL query

8.2.2 LSI-bg

There are different ways of using nearest neighbor in conjunction with Latent Semantic Indexing for the classification (or alternatively the text filtering) task. Our approach has been to use the noisy-or operation in combining the cosine similarity scores of the thirty nearest neighbors to the test example. We have explored the number k slightly, ranging it from 1 to 30. Setting k=1 and choosing the topmost was almost always inferior to the $k = 30$, there was little difference in other numbers. Other combination rules such as k-NN using the majority class, or k-NN that maximizes the sums of the cosine similarities [14, 63] can be explored as well.

Alternatively, we could use Latent Semantic Indexing with other learning paradigms. One approach that we have explored has been that of Local LSI, as described by Hull [30], where a separate reduced space is built for each class. We incorporated the background knowledge into each of these separate spaces, and then re-described the test example into each of these separate spaces as well. The test example was then compared to the training set in each space; whichever it was closest to was returned as the final classification. Our preliminary results did not show that Local LSI was more useful than our initial use of LSI. In our LSI-bg method the background knowledge is used to find factors that more accurately *distinguish* between classes. In Local LSI, since the same set of background knowledge is used in conjunction with each smaller training set, we hoped that the factors that would be found would accurately *describe* each class. However, in our empirical results this did not seem to be the case.

An interesting issue relevant to LSI is that if the training set is small compared to the background text, it may be sufficient to use only the background text, without training data, in X_n. SVD could be performed on the background text alone; both the training and test examples can be re-described in terms of the new space using $\hat{X}_n$. This method of updating SVDs by "folding-in" new documents, has been studied by the LSI community [3]. Although this SVD will not be identical to the one of the training and background examples combined, our initial tests have shown that classification accuracy does not significantly change. This might then provide a mechanism by which incremental learning is achievable — where a new example can be added without requiring a new SVD calculation. Such a method for avoiding the often costly SVD calculation would be one way to manage the otherwise costly work that would be necessary in obtaining new training data in incremental learning scenarios.

8.3 Contributions

We have presented a novel concept of "background knowledge" to aid in text categorization. This is strongly related to the current and growing body of work on the use of unlabeled examples in the text classification task. Rather than having a learner simply rely on labeled training data to create a classifier for new test examples, we show that combining the labeled training data with other forms of available related text allows for the creation of more accurate classifiers. We have incorporated this concept of the addition of background knowledge into three qualitatively different learners, in three very different ways. Yet from the results that we presented, we can see that in many different situations, the use of related data in text classification tasks can provide enough additional information to a learner to reduce error rates on previously unseen examples.

We provide a framework for the incorporation of background knowledge into three distinct text classification learners. In the first approach we show that background knowledge can be used as a set of unlabeled examples in a generative model for text classification. Using the methodology of other researchers that treat the classes of unlabeled examples as missing values, we show that although this background knowledge may be of a different form and type than the training and test sets, it can still be quite useful. Secondly, we view the text classification task as one of information integration using WHIRL, a tool that combines database functionalities with techniques from the information-retrieval literature. We treat the labeled data, test set and background knowledge as three separate databases and use the background knowledge as a bridge to connect elements from the training set to the test set. In this way, training examples are related to a test example in the context of the background knowledge. Lastly, we use Latent Semantic Indexing in conjunction with background knowledge. In this case background knowledge is used with the labeled examples to create a new space in which the training and test examples are re-described. This allows the system to incorporate information from the background knowledge in the similarity comparisons between training and test examples.

References

[1] James Allan, Jamie Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study final report. *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998.

[2] K. Bennet and A. Demiriz. Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, 12:368–374, 1998.

[3] M. Berry, S. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[4] Jeff A. Bilmes. A gentle tutorial of the EM algorithm and its appliation to parameter estimation for Gaussian mixture and hidden Markov models. *Technical Report tr-97-021*, 1998.

[5] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincut. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[6] A. Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

[7] Rebecca Bruce. Semi-supervised learning using prior probabilities and EM. In *Proceedings of the IJCAI01 Workshop on Text Learning: Beyond Supervision*, 2001.

[8] C. Buckley, G. Salton, A. Mitram, and A. Singhal. Automatic query expansion using SMART. In *Proceedings of the Third TExt Retrieval Conference: TREC 3*, 1995.

[9] V. Castelli and T. Cover. On the exponential value of labeled samples. *Pattern Recongnition Letters*, 16:105–111, 1995.

[10] William Cohen. Fast effective rule induction. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1995.

[11] William Cohen. Integration on heterogeneous databases without common domains using queries based on textual similarity. In *Proceedings of ACM-SIGMOD 98*, 1998.

[12] William Cohen. A web-based information system that reasons with structured collections of text. In *Proceedings of Autonomous Agents*, 1998.

[13] William Cohen. Automatically extracting features for concept learning from the Web. In *Proceedings of the 17th International Conference on Machine Learning*, pages 159–166, 2000.

[14] William Cohen and Haym Hirsh. Joins that generalize: Text categorization using WHIRL. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 169–173, 1998.

[15] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural language Processing and Very Large Corpora*, 1999.

[16] F. G. Cozman and I. Cohen. Unlabeled data can degrade classification performance. *Technical Report HPL-2002-234*, 2001.

[17] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Sean Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 509–516, 1998.

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

[19] A. P. Dempster, N. M. Larid, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[20] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[21] S. Dumais. LSI meets TREC: A status report. In D. Hartman, editor, *The first Text REtrieval Conference: NIST special publication 500-215*, pages 105–116, 1993.

[22] S. Dumais. Latent semantic indexing (LSI): TREC-3 report. In D. Hartman, editor, *The Third Text REtrieval Conference, NIST special publication 500-225*, pages 219–230, 1995.

[23] S. Dumais. Combining evidence for effective information filtering. In *AAAI Spring Symposium on Machine Learning and Information Retrieval, Tech Report SS-96-07*, 1996.

[24] S. Dumais, T.A. Letche, M.L. Littman, and T.K. Landauer. Automatic cross-language retrieval using latent semantic indexing. In *AAAI Spring Symposium on Cross-Language Text and Speech Retrieval*, 1997.

[25] P. W. Foltz and S. Dumais. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, 35(12):51–60, 1992.

[26] Jerome H. Friedman. On bias, variance, 0/1 loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.

[27] R. Ghani, S. Slattery, and Yiming Yang. Hypertext categorization using hyperlink patters and meta data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 178–185, 2001.

[28] S. Goldman and Y. Zhou. Enhancing supervised learning with unlabeled data. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[29] P.J. Hayes and S.P Wienstein. Construe/tis: A system for content-based indexing of a database of news stories. In *Proceedings of the Second Annual Conference on Innovative Applications of Artificial Intelligence*, 1990.

[30] David Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *SIGIR94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 282–291, 1994.

[31] Tommi Jaakkola, Marina Meila, and Tony Jebara. Maximum entropy discrimination. In *Advances in Neural Information Processing Systems*, volume 12, 1999.

[32] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98, Tenth European Conference on Machine Learning*, pages 137–142, 1998.

[33] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, 1999.

[34] Daphne Koller and Mehran Sahmi. Hierachically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[35] Ken Lang. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*, pages 331–339, 1995.

[36] Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of Bayesian classifiers. In *National Conference on Artificial Intelligence*, pages 223–228, 1992.

[37] Leah S. Larkey and W. Bruce Croft. Combining classifiers in text categorization. In *SIGIR-96*, 1996.

[38] David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *SIGIR-92*, 1992.

[39] David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *ECML1998: Tenth European Conference on Machine Learning*, 1998.

[40] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 148–156, 1994.

[41] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *SIGIR94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[42] David D. Lewis and K. A. Knowles. Threading electronic mail - a preliminary study. *Information Processing and Management*, 33(2):209–217, 1997.

[43] David D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 3–12, 1994.

[44] Andrew McCallum and Kamal Nigam. Text classification by bootstrapping with keywords, EM and shrinkage. In *ACL '99 Workshop for Unsupervised Learning in Natural Language Processing*, 1999.

[45] Andrew McCallum, Ronald Rosenfeld, Tom Mitchell, and Andrew Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the International Conference of Machine Learning*, 1998.

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

[47] Tom Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[48] Tom Mitchell. The role of unlableled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science*, 1999.

[49] K. Nigam. *Using Unlabeled Data to Improve Text Classification*. PhD thesis, Carnegie Mellon University, 2001.

[50] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, 2000.

[51] Kamal Nigam, Andre Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[52] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[53] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffmann, San Mateo, CA, 1993.

[54] E. Riloff and R. Jones. Learning dictionaries for information extraction using multi-level boot-strapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 474–479, 1999.

[55] S. E. Robertson and K. Spark-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[56] M. Sahmi. A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, 1998.

[57] G. Salton, editor. *Automatic Text Processing*. Addison Welsley, Reading, Massachusetts, 1989.

[58] Martin Szummer and Tommi Jaakkola. Kernel expansions with unlabeled data. In *Advances in Neural Information Processing Systems*, volume 13, 2001.

[59] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[60] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[61] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[62] D. Wu, K. P. Bennet, Nello Cristianini, and John Shawe-Taylor. Large margin trees for induction and transduction. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999.

[63] Y. Yang and C. Chute. An example-based mapping method for text classification and retrieval. *ACM Transactions on Information Systems*, 12, 1994.

[64] S. Zelikovitz and H. Hirsh. Improving short text classification using unlabeled background knowledge to assess document similarity. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1183–1190, 2000.

[65] S. Zelikovitz and H. Hirsh. Using LSI for text classification in the presence of background text. In *Proceedings of the Tenth Conference for Information and Knowledge Management*, 2001.

[66] Tong Zhang and Frank J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Procceedings of the Seventeenth International Conf. on Machine Learning*, pages 1191–1198. Morgan Kaufmann, San Francisco, CA, 2000.

Vita

Sarah Zelikovitz

1985-87	Attended Brooklyn College, Brooklyn, New York.
1988	B.S., Brooklyn College.
1988-89	Graduate work in Computer Science, Brooklyn College, Brooklyn, New York.
1989	M.A. in Computer Science, Brooklyn College.
1989-91	Graduate work in Computer Science, Rutgers, The State University of New Jersey, New Brunswick, New Jersey.
1989-91	Teaching Assistant Department of Computer Science.
1993-2002	Graduate work in Computer Science, Rutgers, The State University of New Jersey, New Brunswick, New Jersey.
1993-99	Teaching Assistant/Lecturer, Department of Computer Science.
2000	Zelikovitz, S. and Hirsh, H. Improving short text classification problems using background knowledge to assess document similarity. *Proceedings of the Seventeenth International Conference on Machine Learning*, 1183-1190.
2001-02	Research Assistant in Computer Science.
2001	Zelikovitz, S. and Hirsh, H. Using LSI for text classification in the presence of background text. *Proceedings of the Tenth Conference for Information and Knowledge Management*, 113-118.
2002	Ph.D. in Computer Science.