Using Unlabeled Data to Improve Text Classification

Kamal Paul Nigam

May 2001

CMU-CS-01-126

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:
Tom M. Mitchell, Chair
Andrew Kachites McCallum
John Lafferty
Tommi Jaakkola, Massachusetts Institute of Technology

This research was sponsored by the National Science Foundation (NSF) under grant no. SBR-9720374, the Defense Advanced Research Projects Agency (DARPA) under contract no. F33615-93-1-1330, and the Digital Equipment Corporation (DEC). The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the NSF, DARPA, the U.S. government or any other entity.

Keywords: text classification, text categorization, unlabeled data, Expectation-Maximization, unsupervised learning

Abstract

One key difficulty with text classification learning algorithms is that they require many hand-labeled examples to learn accurately. This dissertation demonstrates that supervised learning algorithms that use a small number of labeled examples and many inexpensive *unlabeled* examples can create high-accuracy text classifiers. By assuming that documents are created by a parametric generative model, Expectation-Maximization (EM) finds local maximum a posteriori models and classifiers from all the data—labeled and unlabeled. These generative models do not capture all the intricacies of text; however on some domains this technique substantially improves classification accuracy, especially when labeled data are sparse.

Two problems arise from this basic approach. First, unlabeled data can hurt performance in domains where the generative modeling assumptions are too strongly violated. In this case the assumptions can be made more representative in two ways: by modeling sub-topic class structure, and by modeling super-topic hierarchical class relationships. By doing so, model probability and classification accuracy come into correspondence, allowing unlabeled data to improve classification performance. The second problem is that even with a representative model, the improvements given by unlabeled data do not sufficiently compensate for a paucity of labeled data. Here, limited labeled data provide EM initializations that lead to low-probability models. Performance can be significantly improved by using active learning to select high-quality initializations, and by using alternatives to EM that avoid low-probability local maxima.

Acknowledgments

Many different people provided help, support, and input that brought this thesis to fruition. If anyone is looking for an example of a good advisor, let me especially recommend mine, Tom Mitchell. Tom has always been an enthusiastic advisor, providing encouragement, insight, and a valuable big-picture perspective. He has been a shining catalyst for this work. I extend my warmest thanks to Andrew Kachites McCallum. Over the last five years he has been a generous collaborator, a sincere and thoughtful mentor, and a close friend. Some of my fondest memories of graduate school have been our traditional walks to the FedEx box after completing a paper submission. John Lafferty was tremendously helpful in providing his mathematical insight and rigor. In addition, he provided great moral support. I always left my meetings with John feeling uplifted, with a renewed sense of optimism and confidence. I thank Tommi Jaakkola for providing a fresh and external viewpoint on my research. Tommi had the uncanny knack of always asking the questions I had hoped not to hear. By raising these difficult issues, he focused my attention onto critical areas.

I have been fortunate and privileged to work with many different researchers in the field. They have provided me with invaluable knowledge and have shown me what it means to perform careful and responsible research. I thank my collaborators and co-authors: Doug Baker, Mark Craven, Dan DiPasquo, Dayne Freitag, Rayid Ghani, Rosie Jones, John Lafferty, Andrew McCallum, Tom Mitchell, Dunja Mladenic, Sasha Popescul, Choon Quek, Jason Rennie, Ellen Riloff, Kristie Seymore, Sean Slattery, Sebastian Thrun, and Lyle Ungar. My officemates through the years have provided levity and a willing ear on many occasions: Alex Gray, Bryan Loyall, Dimitris Margaritis, Michael Mateas, Eric Tilton, Andrew Tomkins.

This thesis would not have been possible without the love and support of my parents and my sister, Rekha. They were my first teachers and inspired in me a love of learning and a natural curiosity. Last and most importantly, I lovingly thank my wife, Milena. She provides me with the faith and confidence to endure and enjoy it all.

Contents

Chapter 1

Introduction

Suppose we work for a web site that maintains a public listing of job openings from many different companies. A user of the web site might find new career opportunities by browsing all openings in a specific job category. However, these job postings are spidered from the Web, and do not come with any category label. Instead of reading each job post to manually determine the label, it would be helpful to have a system that automatically examines the text and makes the decision itself. This automatic process is called *text classification*. In general, text classification systems categorize documents into one (or several) of a set of pre-defined topics of interest.

Text classification is of great practical importance today given the massive volume of online text available. In recent years there has been an explosion of electronic text from the World Wide Web, electronic mail, corporate databases, chat rooms, and digital libraries. One way of organizing this overwhelming amount of data is to classify it into descriptive or topical taxonomies. For example, Yahoo maintains a large topic hierarchy of web pages. By automatically populating and maintaining these taxonomies, we can aid people in their search for knowledge and information.

How are automatic text classifiers created? Early attempts were based on the manual construction of rule sets. Using this approach a person must compose a detailed set of rules for automatically specifying the class of a document. For example, one such rule might read "If the job posting contains the phrase 'expertise in Java' then the job category is computer programmer." Highly accurate text classifiers were built with this approach, but at significant cost. Constructing a complete rule set requires a lot of domain knowledge and a substantial amount of human time to tune the rules correctly. With few exceptions, this is an impractical approach to text classification.

A more efficient approach is to use supervised learning to construct a classifier. Here, we provide an algorithm with an example set of documents for each class, and allow it to find a representation or decision rule for classifying future documents. This approach also gives high-accuracy classifiers, and is significantly less expensive than manual construction because the algorithm automatically constructs the decision rule itself. Supervised text classification algorithms have been successfully used in a wide variety of practical domains. A few examples are: cataloging news articles (Lewis & Gale, 1994; Joachims, 1998) and web pages (Craven et al., 2000; Shavlik & Eliassi-Rad, 1998), learning the reading interests of users (Pazzani et al., 1996; Lang, 1995), and sorting electronic mail (Lewis & Knowles, 1997; Sahami et al., 1998).

However, the supervised learning approach is not as effortless as we might hope. One key difficulty with these algorithms is that they require a large, often prohibitive, number of labeled training examples to learn accurately. Labeling must typically be done by a person; this is a painfully time-consuming process. Take, for example, the task of learning which newsgroup articles are of interest to a particular person reading UseNet news. Work by Lang (1995) found that after a person read and hand-labeled about 1000 articles, a learned classifier achieved a precision of about 50% when making predictions for only the top 10% of documents about which it was most confident. Most users of a practical system, however, would not have the patience to label a thousand articles—especially to obtain only this level of precision. One would obviously prefer algorithms that can provide accurate classifications after hand-labeling only a dozen articles, rather than thousands. This need for large quantities of expensive labeled examples raises an important question: what other sources of information can reduce the need for labeled data?

The goal of this thesis is to demonstrate that supervised learning algorithms using a small number of labeled examples and a large number of *unlabeled* examples create high-accuracy text classifiers. In general, unlabeled examples are much less expensive and easier to come by than labeled examples. This is particularly true for text classification tasks involving online data sources, such as web pages, email, and news stories, where huge amounts of unlabeled text are readily available. Collecting this text can frequently be done automatically, so it is feasible to quickly gather a large set of unlabeled examples. If unlabeled data can be integrated into supervised learning then building text classification systems will be significantly faster and less expensive than before.

1.1 Unlabeled data? Are we crazy?

At first glance, it might seem that nothing is to be gained from unlabeled data. After all, an unlabeled document doesn't contain the most important piece of information—its class. Here is an intuitive example of how unlabeled data might be useful. Suppose we are interested in recognizing web pages about academic courses. We are given just a few known course and non-course web pages, along with a large number of web pages that are unlabeled. By looking at just the labeled data we determine that pages containing the word *homework* tend to be about academic courses. If we use this fact to estimate the classification of the many unlabeled web pages, we might find that the word *lecture* occurs frequently in the unlabeled examples that are now believed to belong to the positive class. This co-occurrence of the words *homework* and *lecture* over the large set of unlabeled training data can provide useful information to construct a more accurate classifier that considers both *homework* and *lecture* as indicators of positive examples. In this thesis, we explain how unlabeled data can be used to increase classification accuracy, especially when labeled data are scarce.

Formally, what information do unlabeled data provide? By themselves they give us knowledge only of the distribution of examples in feature space. In the most general case, distributional knowledge will not provide helpful information to supervised learning. Consider classifying uniformly distributed instances based on conjunctions of literals. Here there is no relationship between the uniform instance distribution and the space of possible classification tasks and clearly, unlabeled data can not help. We need to introduce appropriate bias into our learner by assuming some dependence between the instance distribution and the classification task.

Even standard supervised learning with labeled data must introduce bias in order to learn. In a well-known and often-proven result, Watanabe's (1969) Theorem of the Ugly Duckling shows that without bias all pairs of training examples look equally similar, and generalization into classes is impossible. This result was foreshadowed long ago by both William of Ockham's philosophy of radical nominalism in the 1300's and David Hume in the 1700's. Somewhat more recently, Zhang and Oles (2000) formalized and proved that supervised learning with unlabeled examples must assume a dependence between the instance distribution and the classification task.

In this thesis we assume the dependence can be captured by a parametric generative model for text documents and their class labels—a statistical process that creates the words and the class of a document. For example, our generative model encodes which words are more common in one class than another. Using this, it creates a document in a given class by randomly selecting words according to the class's word frequencies. We know this process is not realistic, and that statistical models will not capture the subtleties of the authoring process. Nevertheless, these assumptions encode a relationship between the document distribution and the classification task that allow unlabeled data to be incorporated into learning.

For a specific classification task, we select the model's parameter values using the evidence contained in the labeled and unlabeled data. Given a parameterization of a model, we can judge how probable it was that that model generated our data. We learn a text classifier by searching for the parameter setting that would be the most probable to have generated the labeled and unlabeled data. For this we use the statistical technique Expectation-Maximization (EM) (Dempster et al., 1977). While it typically does not find the most probable parameter values, it does find a local maxima in parameter space. Once EM gives us a parameter setting, we can use the generative model for text classification. Given a test document without a class label, we evaluate which class label was most likely to have been generated for the words in the given document. Our hope is that highly probable generative models found using unlabeled data correspond to high-accuracy text classifiers, even when our models are not perfect.

1.2 Questions asked

This thesis asks and answers four central questions:

Can a generative model approach use unlabeled data to improve text classification?

The types of generative models we propose to use for text classification are simple in comparison to the complexity of human authoring. They maintain no sense of syntax, context, or even word order. It's certainly not obvious that maximizing the probability of an imperfect model will increase classification accuracy. Will our models be representative enough for the purposes of text classification? There is some reason to be initially pessimistic. Similar generative model approaches for related text tasks, such as part-of-speech tagging (Merialdo, 1994; Elworthy, 1994) and information extraction (Seymore et al., 1999) have shown that incorporating unlabeled data into supervised learning decreases performance of these systems. However, text classification is not as dependent as these tasks on model correctness at a local level. We demonstrate that generative models are a helpful and valid approach to using unlabeled data for text classification.

Can all text classification domains use the same generative model?

Different text classification domains very considerably in their properties. Some have only very short documents; others have quite disparate lengths. Some tasks are easy and well-understood; others are complex and ill-defined. Some classes are narrow; others are multi-faceted. With such a wide variety of text, it is appropriate to wonder whether all tasks can successfully use unlabeled data with the same generative model class. We demonstrate that for some domains, unlabeled data is easily used with the most straightforward generative model suggested. For other tasks, incorporating unlabeled data with the basic model lowers classification accuracy.

Can we adapt our basic generative model to use unlabeled data on more challenging text classification domains?

For unlabeled data to be useful in a generative model approach, classification accuracy and model likelihood should be correlated. With a perfect model and sufficient unlabeled data this is a given, but with our simple models of text there are no guarantees. When our modeling assumptions do not approximate the true data distribution well enough, unlabeled data will not improve classification. In these cases, it may be possible to change or relax our assumptions to more closely match the data. We adapt our models in two different directions by modeling the sub-topic structure of a class, and the super-topic hierarchical class relationships. These adapted models allow unlabeled data to be useful when learning text classifiers for these more difficult domains.

Can we improve accuracy even further when labeled data are sparse?

When a generative model needs no adaptation unlabeled data improve accuracy significantly, but even then they can not compensate for an extreme lack of labeled data. In these cases, the EM maximization process is unable to find highly probable parameters. Viewing EM as an iterative hill-climbing algorithm, the initialization given by sparse labeled data are the source of the trouble. We confront this weakness in two ways: by selecting documents for labeling that provide higher-quality initializations, and by using other likelihood maximization techniques that are insensitive to poor initialization.

1.3 Road map

The outline of this thesis is as follows. The technical content is contained in the next three chapters. Chapter 2 presents the baseline generative model for text documents and classes. It derives the learning algorithm that incorporates unlabeled data into supervised learning by likelihood maximization with EM. We show promising experimental results showing significant benefit for using unlabeled data on some datasets. For other datasets, this basic approach is inadequate and unlabeled data can hurt classification accuracy.

Chapter 3 explores how to change the modeling assumptions in response to poor performance of unlabeled data using the baseline model. For domains with multifaceted, complex classes, the sub-topic structure of a class can be more accurately modeled by relaxing our assumption of a one-to-one correspondence between mixture components and classes. For domains with a hierarchy indicating class similarities and relationships, the super-topic structure can be more accurately modeled by relaxing the assumptions about the independence between classes. Both of these enhancements allow unlabeled data to improve text classification accuracy on domains unresponsive to the basic approach.

Chapter 4 improves the performance of classifiers built with unlabeled data and sparse labeled data. After an analysis of the effects of the limited labeled data, we demonstrate that its use for EM initialization hinders performance. We improve accuracy by using a Query-By-Committee active learning approach to select high-quality labeled documents for initialization. We also show that deterministic annealing, a likelihood maximization technique similar to EM, is less sensitive to poor initialization and finds more accurate text classifiers.

Chapter 5 surveys the current state of text classification. It also relates different approaches to combining labeled and unlabeled data, and contrasts them with the approach taken in this thesis. Chapter 6 discusses the conclusions of this dissertation and the answers to the questions already posed. It also proposes several avenues for further research suggested by our findings.

Chapter 2

Incorporating Unlabeled Data with Generative Models

One way to incorporate unlabeled data into supervised learning is to assume that a statistical process generates the documents. By precisely specifying the generative model, we can use the statistical technique Expectation-Maximization to find high-probability parameters of the model given a combination of labeled and unlabeled data. This model can then be turned around and used for text classification, since the generative model has an embedded notion of class. Experimental evidence shows that using unlabeled data with Expectation-Maximization can increase classification accuracy. Additional evidence shows, however, that this straightforward approach does not always perform well. There are two hypotheses that could explain this: (1) the assumptions made about the generative process result in an unrepresentative model, so that model probability and classification accuracy are not well-correlated, or (2) the Expectation-Maximization search suffers from getting stuck in local maxima with low-probability models. The remainder of this thesis shows how to overcome these problems.

2.1 Introduction

In this dissertation, we appeal to the framework of statistical modeling. Even though text is written by people with a strong and complicated set of rules governing composition and grammar, we will specify a simple statistical generative model for the production of text documents. Implicit in this model are the assumptions that (1) documents are generated from class-specific probability distributions, and (2) words occur in documents independently of each other given the class. With this model it is straightforward to find the most likely parameters given a set of labeled data. Despite the oversimplifications in the generative process, practitioners have successfully used this naive Bayes approach in text classification for many years. We begin by taking the same approach, but with a combination of labeled and unlabeled data. Using the same statistical model as naive Bayes, we add the evidence of unlabeled data when finding high-probability generative parameters for our assumed model.

We introduce an algorithm for learning from labeled and unlabeled documents based on the combination of Expectation-Maximization (EM) and a naive Bayes classifier. EM is an iterative technique for maximum likelihood or maximum a posteriori parameter estimation in problems with incomplete data (Dempster et al., 1977). In our case, the unlabeled data are considered incomplete because they come without class labels. The algorithm first trains a classifier with only the available *labeled* documents, and assigns probabilistically-weighted class labels to each unlabeled document by using the classifier to calculate their expectation. It then trains a new classifier using all the documents—both the originally labeled and the formerly unlabeled—and iterates. EM performs hill-climbing in model probability space, finding the classifier parameters that locally maximize the probability of the model given all the data—both the labeled and the unlabeled. We combine EM with naive Bayes, a classifier based on a mixture of multinomials, that is commonly used in text classification.

Experimental results, obtained using text from two different real-world tasks, show that using unlabeled data reduces classification error by up to 30%. For a fixed classification error unlabeled data dramatically reduce the number of labeled examples needed. For example, to identify the source newsgroup for a UseNet article with 70% classification accuracy, a traditional learner requires 2000 labeled examples; alternatively our algorithm takes advantage of 10000 unlabeled examples and requires only 600 labeled examples to achieve the same accuracy. Thus, in this task, the technique reduces the need for labeled training examples by more than a factor of three. With only 40 labeled documents (two per class), accuracy is improved from 27% to 43% by adding unlabeled data. These findings illustrate the power of unlabeled data in text classification problems, and also demonstrate the strength of the algorithms proposed here.

However, on a different dataset, we show that basic EM can suffer from a misfit between the modeling assumptions and the unlabeled data. These results raise a number of interesting questions that motivate the remainder of the dissertation.

The outline of this chapter is as follows. Section 2.2 specifies the naive Bayes generative model, the baseline used throughout this thesis. Section 2.3 discusses model probability as an optimization criteria for choosing parameter settings of the generative model. Section 2.4 explains how to optimize model likelihood with a set of labeled data and how to use a model for text classification purposes. Section 2.5 presents how to locally optimize model likelihood given a combination of labeled and unlabeled data. Section 2.6 presents experimental results showing that using a combination of labeled and unlabeled data can outperform the use of labeled data alone. Section 2.7 discusses the findings and poses the challenges and questions addressed by the rest of the dissertation.

2.2 A Generative Model for Text

This section presents a probabilistic framework for characterizing the nature of documents and classifiers. The framework defines a probabilistic generative model for the data, and embodies three assumptions about the generative process: (1) the data are produced by a mixture model, (2) there is a one-to-one correspondence between mixture components and classes, and (3) the mixture components are multinomial distributions of individual words—the words of a document are produced independently of each other given the class. From these assumptions we can derive the naive Bayes classifier, a simple and commonly-used tool for text categorization, by finding the most probable parameters for the model.

Documents are generated by a *mixture of multinomials* model, where each mixture component corresponds to a class. Let there be $|C|$ classes and a vocabulary of size $|V|$; each document d has $|d|$ words in it. How do we create a document using this model? First, we roll a biased $|C|$-sided die to determine the class of our document. Then, we pick up the biased $|V|$-sided die that corresponds to the chosen class. We roll this die $|d|$ times, and write down the indicated words. These words form the generated document.

Formally, every document is generated according to a probability distribution defined by the parameters for the mixture model, denoted θ. The probability distribution consists of a mixture of components $c_j \in \mathcal{C} = \{c_1, ..., c_{|\mathcal{C}|}\}$. Each component is parameterized by a disjoint subset of θ. A document, d_i, is created by first selecting a mixture component according to the mixture weights (or class probabilities), $\mathrm{P}(c_j|\theta)$, then having this selected mixture component generate a document according to its own parameters, with distribution $\mathrm{P}(d_i|c_j;\theta)$.[1] Thus, we can characterize the likelihood of document d_i with a sum of total probability over all mixture components:

$$\mathrm{P}(d_i|\theta) = \sum_{j=1}^{|\mathcal{C}|} \mathrm{P}(c_j|\theta)\mathrm{P}(d_i|c_j;\theta). \tag{2.1}$$

Each document has a class label. We assume that there is a one-to-one correspondence between mixture model components and classes, and thus use c_j to indicate the jth mixture component, as well as the jth class. The class label for a particular document d_i is written y_i. If document d_i was generated by mixture component c_j we say $y_i = c_j$. The class label may or may not be known for a given document.

A document, d_i, is considered to be an ordered list of word events, $\langle w_{d_{i,1}}, w_{d_{i,2}}, \ldots \rangle$. We write $w_{d_{i,k}}$ for the word w_t in position k of document d_i, where w_t is a word in the vocabulary $V = \langle w_1, w_2, \ldots, w_{|V|} \rangle$. For example, if this paragraph was document 7, then $w_{d_{7,2}}$ would be the word *document*. When a document is to be generated by a particular mixture component a document length, $|d_i|$, is first chosen independently of the component. (Note that this assumes that document length is independent of class.[2]) Then, the selected mixture component generates a word sequence of the specified length. We assume it generates each word independently of the length.

Thus, we can expand the second term from Equation 2.1, and express the probability of a document given a mixture component in terms of its constituent features: the document length and the words in the document. Note that, in this general setting, the probability of a word event must be conditioned on all the words that precede it.

$$\mathrm{P}(d_i|c_j;\theta) = \mathrm{P}(\langle w_{d_{i,1}}, \ldots, w_{d_{i,|d_i|}} \rangle | c_j;\theta) = \mathrm{P}(|d_i|) \prod_{k=1}^{|d_i|} \mathrm{P}(w_{d_{i,k}}|c_j;\theta; w_{d_{i,q}}, q < k) \tag{2.2}$$

[1] We use standard notational shorthand for random variables, whereby $\mathrm{P}(X = x_i|Y = y_j)$ is written $\mathrm{P}(x_i|y_j)$ for random variables X and Y taking on values x_i and y_j.

[2] Previous naive Bayes formalizations do not include this document length effect. In the most general case, document length should be modeled and parameterized on a class-by-class basis.

Next we make the standard naive Bayes assumption: that the words of a document are generated independently of context, that is, independently of the other words in the same document given the class label. We further assume that the probability of a word is independent of its position within the document; thus, for example, the probability of seeing the word *homework* in the first position of a document is the same as seeing it in any other position. We can express these assumptions as:

$$\mathrm{P}(w_{d_{i,k}}|c_j;\theta;w_{d_{i,q}},q<k)=\mathrm{P}(w_{d_{i,k}}|c_j;\theta). \tag{2.3}$$

Combining these last two equations gives the naive Bayes expression for the probability of a document given its class:

$$\mathrm{P}(d_i|c_j;\theta)=\mathrm{P}(|d_i|)\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{i,k}}|c_j;\theta). \tag{2.4}$$

Thus the parameters of an individual mixture component define a multinomial distribution over words,[3] *i.e.* the collection of word probabilities, each written $\theta_{w_t|c_j}$, such that $\theta_{w_t|c_j}\equiv\mathrm{P}(w_t|c_j;\theta)$, where $t=\{1,\ldots,|V|\}$ and $\sum_t \mathrm{P}(w_t|c_j;\theta)=1$. Since we assume that for all classes, document length is identically distributed, it does not need to be parameterized for classification. The only other parameters of the model are the mixture weights (class probabilities), written θ_{c_j}, which indicate the probabilities of selecting the different mixture components. Thus the complete collection of model parameters, θ, defines a set of multinomials and class probabilities: $\theta=\{\theta_{w_t|c_j}:w_t\in V,c_j\in\mathcal{C}\ ;\ \theta_{c_j}:c_j\in\mathcal{C}\}$.

2.3 Model Probability as an Optimization Criteria

The ultimate goal of building a text classifier is to find one that will have high accuracy on previously unseen examples. Since we don't have these examples on hand with their labels, we must find an alternative criteria to use when selecting classifier parameters.

[3] The astute statistician will note that the multinomial coefficients are missing and this is not truly a multinomial distribution. This is because we have put a word order into our generative model. A real mixture of multinomials uses no order, but gives exactly the same classifiers, as the coefficients cancel out (McCallum & Nigam, 1998a).

Several criteria have been described in the literature, all with a reasonable amount of success when given a large set of labeled examples.

The first potential criteria is to maximize the classification accuracy on the labeled examples at hand. This has been done with various gradient descent approaches (Lewis et al., 1996). One problem with this approach is that it is quite easy to find parameters that maximize classification accuracy on the labeled set, without getting good generalization for future examples. This is because the dimensionality of text classification is so large that it is easy to overfit the training data. Additionally, there are many possible solutions that correctly classify the labeled data, so it is not clear which to choose. Finally, it might be hard to imagine using unlabeled data with such an optimization criteria, because without labels these examples do not play a role in estimating classification accuracy.

A second criteria is to maximize the margin between the different classes. This approach, used by Support Vector Machines, for example, is to find the linear separator between the classes that is furthest away from the data. This approach tends to find separators that lie in valleys of low document density. This approach has been adapted to work with labeled and unlabeled data (Joachims, 1999) and has shown promising results. Intuitively, a classification boundary that separates the data and is also far away from the data provides a method for achieving low classification error.

The presence of the assumed generative model in our scenario gives us a third way to build a classifier. The approach that we will take is to maximize the probability of the model given the training data. Since we are using a setting with a strong probabilistic interpretation, it is very natural to talk about the probability of a model parameterization given a set of data. If the generative model were correct, then with an infinite amount of data the most probable solution would indeed give us the solution that maximizes classification accuracy. When we don't necessarily believe the generative model, it is an interesting question to consider whether maximizing model probability is a reasonable optimization criteria.

2.4 Naive Bayes Text Classification

This section presents the naive Bayes text classifier. It is traditionally trained using a collection of labeled documents. Naive Bayes finds the most probable parameters for the statistical model introduced in Section 2.2 given a set of labeled data. Naive

Bayes is the foundation upon which we will later build when integrating unlabeled data into supervised learning.

2.4.1 Training a Naive Bayes Classifier with Labeled Data

Learning a naive Bayes text classifier consists of estimating the parameters of the generative model by using a set of labeled training data, $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\}$. The estimate of the parameters θ is written $\hat{\theta}$. Naive Bayes uses the maximum a posteriori (MAP) estimate, thus finding $\arg\max_\theta \mathrm{P}(\theta|\mathcal{D})$. This is the value of θ that is most probable given the evidence of the training data and a prior.

To perform MAP estimation we must specify the prior distribution over our model space. Our prior distribution is formed with the product of Dirichlet distributions—one for each class multinomial and one for the overall class probabilities. The Dirichlet is a commonly-used conjugate prior distribution over multinomials. The form of the Dirichlet is:

$$\mathrm{P}(\theta_{w_t|c_j}) \propto \prod_{t=1}^{|V|} \mathrm{P}(w_t|c_j)^{\alpha_t - 1} \tag{2.5}$$

where the α_t are constants greater than zero. We set all $\alpha_t = 2$, which corresponds to a prior that favors the uniform distribution. A well-presented introduction to Dirichlet distributions is given by Stolcke and Omohundro (1994).

The parameter estimation formulae that result from maximization with the data and our prior are the familiar ratios of empirical counts. The estimated probability of a word given a class, $\hat{\theta}_{w_t|c_j}$, is simply the number of times word w_t occurs in the training data for class c_j, divided by the total number of word occurrences in the training data for that class—where counts in both the numerator and denominator are augmented with pseudo-counts (one for each word) that come from the Dirichlet prior over each $\theta_{w_t|c_j}$. The use of this type of prior is sometimes referred to as *Laplace smoothing*. Smoothing is necessary to prevent zero probabilities for infrequently occurring words.

The word probability estimates $\hat{\theta}_{w_t|c_j}$ are:

$$\hat{\theta}_{w_t|c_j} \equiv \mathrm{P}(w_t|c_j;\hat{\theta}) = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} z_{ij} N(w_t, d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|\mathcal{D}|} z_{ij} N(w_s, d_i)}, \tag{2.6}$$

where $N(w_t, d_i)$ is the count of the number of times word w_t occurs in document d_i and where z_{ij} is given by the class label: 1 when $y_i = c_j$ and otherwise 0.

The class probabilities, $\hat{\theta}_{c_j}$, are estimated in the same manner, and also involve a ratio of counts with smoothing:

$$\hat{\theta}_{c_j} \equiv \mathrm{P}(c_j|\hat{\theta}) = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} z_{ij}}{|\mathcal{C}| + |\mathcal{D}|}. \tag{2.7}$$

The derivation of these ratios-of-counts formulae comes directly from maximum a posteriori parameter estimation. Finding the θ that maximizes $\mathrm{P}(\theta|\mathcal{D})$ is accomplished by first breaking this expression into two terms by Bayes' rule: $\mathrm{P}(\theta|\mathcal{D}) \propto \mathrm{P}(\mathcal{D}|\theta)\mathrm{P}(\theta)$. The first term is calculated by the product of all the document likelihoods (from Equation 2.1). The second term, the prior distribution over parameters, is the product of Dirichlets. The whole expression is maximized by solving the system of partial derivatives of $\log(\mathrm{P}(\theta|\mathcal{D}))$, using Lagrange multipliers to enforce the constraint that the word probabilities in a class must sum to one. This maximization yields the ratio of counts seen above.

2.4.2 Classifying New Documents with Naive Bayes

Given estimates of these parameters calculated from the training documents according to Equations 2.6 and 2.7, it is possible to turn the generative model backwards and calculate the probability that a particular mixture component generated a given document. We derive this by an application of Bayes' rule, and then by substitutions using Equations 2.1 and 2.4:

$$\begin{aligned} \mathrm{P}(y_i = c_j|d_i;\hat{\theta}) &= \frac{\mathrm{P}(c_j|\hat{\theta})\mathrm{P}(d_i|c_j;\hat{\theta})}{\mathrm{P}(d_i|\hat{\theta})} \\ &= \frac{\mathrm{P}(c_j|\hat{\theta}) \prod_{k=1}^{|d_i|} \mathrm{P}(w_{d_{i,k}}|c_j;\hat{\theta})}{\sum_{r=1}^{|\mathcal{C}|} \mathrm{P}(c_r|\hat{\theta}) \prod_{k=1}^{|d_i|} \mathrm{P}(w_{d_{i,k}}|c_r;\hat{\theta})}. \end{aligned} \tag{2.8}$$

If the task is to classify a test document d_i into a single class, then the class with the highest posterior probability, $\arg\max_j \mathrm{P}(y_i = c_j|d_i;\hat{\theta})$, is selected.

Note that all the assumptions about the generation of text documents (mixture model, one-to-one correspondence between mixture components and classes, word independence, and document length distribution) are violated in real-world text data. Documents are often mixtures of multiple topics. Words within a document are not independent of each other—grammar and topicality make this so.

Despite these violations, empirically the Naive Bayes classifier does a good job of classifying text documents (Lewis & Ringuette, 1994; Craven et al., 2000; Yang & Pedersen, 1997; Joachims, 1997; McCallum et al., 1998). This observation is explained in part by the fact that classification estimation is only a function of the sign (in binary classification) of the function estimation (Domingos & Pazzani, 1997; Friedman, 1997). The word independence assumption causes naive Bayes to give extreme (almost 0 or 1) class probability estimates. However, these estimates can still be poor while classification accuracy remains high.

2.5 Learning a Naive Bayes Model from Labeled and Unlabeled Data

In the previous section we showed how to find maximum a posteriori parameter estimates given a set of labeled data. With labeled and unlabeled data, we would like to similarly find MAP parameters. Because there are no labels for the unlabeled data, the closed-form equations from the previous section are not applicable. However, using the Expectation-Maximization (EM) technique from statistics, we can find locally MAP parameter estimates for the same generative model.

The EM technique as applied to the case of labeled and unlabeled data with naive Bayes yields a straightforward and appealing algorithm. A schematic of this algorithm is shown in Figure 2.1. First, a naive Bayes classifier is built in the standard supervised fashion from the limited amount of labeled training data. Then, we perform classification of the unlabeled data with the naive Bayes model. We note not just the most likely class but the probabilities associated with each class. Then, we rebuild a new naive Bayes classifier using all the data—labeled and unlabeled—using the estimated class probabilities as true class labels. This means that the unlabeled documents are treated as several fractional documents according to the class probabilities. We iterate this process of classifying the unlabeled data and rebuilding the naive Bayes model until it converges to a stable classifier and set of labels for the data. The statistical foundation for this algorithm is described in the next section.

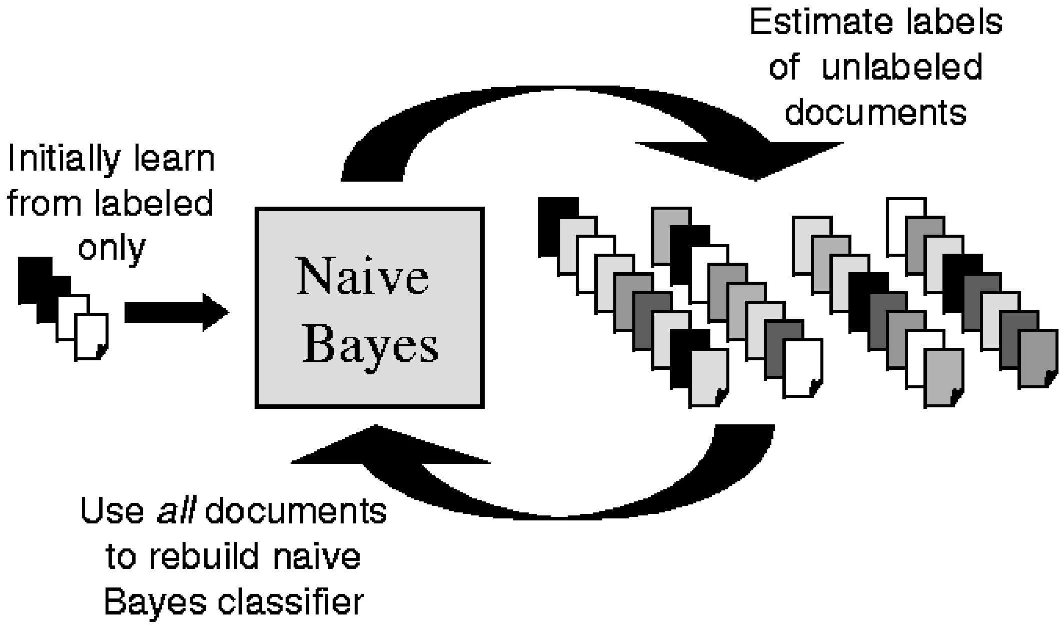
Estimate labels of unlabeled documents
Initially learn from labeled only
Naive Bayes
Use *all* documents to rebuild naive Bayes classifier

Figure 2.1: A schematic for building text classifiers from labeled and unlabeled data with EM.

2.5.1 Expectation-Maximization

We are given a set of training documents $\mathcal{D}$ and the task is to build a classifier in the form of the previous section. However, unlike previously, only some subset of the documents $d_i \in \mathcal{D}^l$ come with class labels $y_i \in \mathcal{C}$, and the rest of the documents, in subset $\mathcal{D}^u$, come without class labels. Thus we have a disjoint partitioning of $\mathcal{D}$, such that $\mathcal{D} = \mathcal{D}^l \cup \mathcal{D}^u$.

As in Section 2.4.1, learning a classifier is approached as calculating a maximum a posteriori estimate of θ, *i.e.* $\arg\max_\theta \mathrm{P}(\theta)\mathrm{P}(\mathcal{D}|\theta)$. Consider the second term of the maximization, the probability of all the training data, $\mathcal{D}$. The probability of all the data is simply the product of the probabilities of each document, because documents are independent of each other, given the model. For the unlabeled data, the probability of an individual document is a sum of total probability over all the classes, as in Equation 2.1. For the labeled data, the generating component is already given by labels y_i and we do not need to refer to all mixture components—just the one corresponding to the class. Thus, the probability of all the data is:

- **Inputs:** Collections $\mathcal{D}^l$ of labeled documents and $\mathcal{D}^u$ of unlabeled documents.
- Build an initial naive Bayes classifier, $\hat{\theta}$, from the labeled documents, $\mathcal{D}^l$, only. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \arg\max_\theta \mathrm{P}(\mathcal{D}|\theta)\mathrm{P}(\theta)$ (see Equations 2.6 and 2.7).
- Loop while classifier parameters improve, as measured by the change in $l_c(\theta|\mathcal{D};\mathbf{z})$ (the complete log probability of the labeled and unlabeled data, and the prior) (see Equation 2.11):
    - **(E-step)** Use the current classifier, $\hat{\theta}$, to estimate component membership of each unlabeled document, *i.e.*, the probability that each mixture component (and class) generated each document, $\mathrm{P}(c_j|d_i;\hat{\theta})$ (see Equation 2.8).
    - **(M-step)** Re-estimate the classifier, $\hat{\theta}$, given the estimated component membership of each document. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \arg\max_\theta \mathrm{P}(\mathcal{D}|\theta)\mathrm{P}(\theta)$ (see Equations 2.6 and 2.7).
- **Output:** A classifier, $\hat{\theta}$, that takes an unlabeled document and predicts a class label.

Table 2.1: The basic EM algorithm described in Section 2.5.

$$\begin{aligned}\mathrm{P}(\mathcal{D}|\theta) &= \prod_{d_i \in \mathcal{D}^u} \sum_{j=1}^{|\mathcal{C}|} \mathrm{P}(c_j|\theta)\mathrm{P}(d_i|c_j;\theta) \\ &\quad \times \prod_{d_i \in \mathcal{D}^l} \mathrm{P}(y_i = c_j|\theta)\mathrm{P}(d_i|y_i = c_j;\theta). \qquad (2.9)\end{aligned}$$

Instead of trying to maximize $\mathrm{P}(\theta|\mathcal{D})$ directly we work with $\log(\mathrm{P}(\theta|\mathcal{D}))$ instead; a maximum in one is a maximum in the other. Using Equation 2.9, we write:

$$\begin{aligned}\log \mathrm{P}(\theta|\mathcal{D}) &= \log(\mathrm{P}(\theta)) + \sum_{d_i \in \mathcal{D}^u} \log \sum_{j=1}^{|\mathcal{C}|} \mathrm{P}(c_j|\theta)\mathrm{P}(d_i|c_j;\theta) \\ &\quad + \sum_{d_i \in \mathcal{D}^l} \log\left(\mathrm{P}(y_i = c_j|\theta)\mathrm{P}(d_i|y_i = c_j;\theta)\right). \qquad (2.10)\end{aligned}$$

We call this the *incomplete* log probability, because the unlabeled data are not complete without labels. (We have dropped the constant term $\log \mathrm{P}(\mathcal{D})$ for convenience.) Notice that this equation contains a log of sums for the unlabeled data, which makes a maximization by partial derivatives computationally intractable. But, if we had access to the class labels of all the documents (the set $\mathbf{z}$ containing all binary indicators z_{ij}) then we could express the *complete* log probability of the parameters, $\log \mathrm{P}_c(\theta|\mathcal{D}, \mathbf{z})$, without a log of sums, because only one term inside the sum would be non-zero.

$$\log \mathrm{P}_c(\theta|\mathcal{D}; \mathbf{z}) = \log(\mathrm{P}(\theta)) + \sum_{d_i \in \mathcal{D}} \sum_{j=1}^{|\mathcal{C}|} z_{ij} \log \left(\mathrm{P}(c_j|\theta)\mathrm{P}(d_i|c_j; \theta)\right) \qquad (2.11)$$

If we replace z_{ij} for the unlabeled documents by its expected value according to the current model $(\mathrm{P}(c_j|d_i, \hat{\theta}))$, then Equation 2.11 bounds from below the incomplete log probability from Equation 2.10. This can be shown by an application of Jensen's inequality (*e.g.* $E[\log(X)] \geq \log(E[X])$). As a result one can find a locally maximum $\hat{\theta}$ by a hill climbing procedure. This was formalized as the *Expectation-Maximization* (EM) technique and proven by Dempster et al. (1977).

The iterative hill climbing procedure alternately recomputes the expected value of $\mathbf{z}$ and the maximum a posteriori parameters given the expected value of $\mathbf{z}$. Note that for the labeled documents z_{ij} is already known. It must, however, be estimated for the unlabeled documents. Let $\hat{\mathbf{z}}^{(k)}$ and $\hat{\theta}^{(k)}$ denote the estimates for $\mathbf{z}$ and θ at iteration k. Then, the algorithm finds a local maximum of $l(\theta|\mathcal{D})$ by iterating the following two steps:

- E-step: Set $\hat{\mathbf{z}}^{(k+1)} = E[\mathbf{z}|\mathcal{D}; \hat{\theta}^{(k)}]$.
- M-step: Set $\hat{\theta}^{(k+1)} = \arg\max_{\theta} \mathrm{P}(\theta|\mathcal{D}; \hat{\mathbf{z}}^{(k+1)})$.

In practice, the E-step corresponds to calculating probabilistic labels $\mathrm{P}(c_j|d_i; \hat{\theta})$ for the unlabeled documents by using the current estimate of the parameters, $\hat{\theta}$, and Equation 2.8. The M-step, maximizing the complete likelihood equation, corresponds to calculating a new maximum a posteriori estimate for the parameters, $\hat{\theta}$, using the current estimates for $\mathrm{P}(c_j|d_i; \hat{\theta})$, and Equations 2.6 and 2.7.

Any initialization of the parameters will lead to some local maxima with EM. Many instantiations of EM begin by choosing a starting model parameterization randomly.

In our case, we can be more selective about the starting point since we have not only unlabeled data, but also some labeled data. Our iteration process is initialized with a priming M-step, in which only the labeled documents are used to estimate the classifier parameters, $\hat{\theta}$, as in Equations 2.6 and 2.7. Then the cycle begins with an E-step that uses this classifier to probabilistically label the unlabeled documents for the first time.

The algorithm iterates until it converges to a point where $\hat{\theta}$ does not change from one iteration to the next. Algorithmically, we determine that convergence has occurred by observing a below-threshold change in the log-probability of the parameters (Equation 2.11), which is the height of the surface on which EM is hill-climbing.

Table 2.1 gives an outline of the basic EM algorithm from this section.

2.5.2 Discussion

In summary, EM finds a $\hat{\theta}$ that locally maximizes the posterior probability of its parameters given *all* the data—both the labeled and the unlabeled. It provides a method whereby unlabeled data can augment limited labeled data and contribute to parameter estimation. An interesting empirical question is whether these more probable parameter estimates will improve classification accuracy. Section 2.4.2 discussed the fact that naive Bayes usually performs classification well despite violations of its assumptions. Will this still hold true when using unlabeled data?

Note that the justifications for this approach depend on the assumptions stated in Section 2.2, namely, that the data are produced by a mixture model, and that there is a one-to-one correspondence between mixture components and classes. If the generative modeling assumptions were correct, then maximizing model probability would be a good criteria indeed. The Bayes optimal classifier corresponds to the MAP parameter estimates of the model. When these assumptions do not hold—as certainly is the case in real-world textual data—the benefits of unlabeled data are less clear. The next section will show empirically that this method can indeed dramatically improve the accuracy of a document classifier, especially when there are only a few labeled documents.

Expectation-Maximization is a well-known family of algorithms with a long history and many applications. Its application to classification is not new in the statistics literature. The idea of using an EM-like procedure to improve a classifier with unlabeled data predates the formulation of EM (e.g., Titterington, 1976). A survey of this related work is given in Section 5.2.1. Our application of EM with a mixture of multinomials is the first application of this approach to text classification.

2.6 Experimental Results

In this section, we provide empirical evidence that combining labeled and unlabeled training documents using EM outperforms traditional naive Bayes, which trains on labeled documents alone. We present experimental results with three different text corpora: UseNet newsgroups (20 Newsgroups), web pages (WebKB), and newswire articles (Reuters).[4] Results show that improvements in accuracy due to unlabeled data are often dramatic, especially when the number of labeled training documents is low. For example, on the 20 Newsgroups data set, classification error is reduced by 30% when trained with 300 labeled and 10000 unlabeled documents.

2.6.1 Datasets and Protocol

The 20 Newsgroups data set (Mitchell, 1997; Joachims, 1997; McCallum et al., 1998), collected by Ken Lang, consists of 20017 articles divided almost evenly among 20 different UseNet discussion groups. The task is to classify an article into the one newsgroup (of twenty) to which it was posted. Many of the categories fall into confusable clusters; for example, five of them are `comp.*` discussion groups, and three of them discuss religion. When words from a stoplist of common short words are removed, there are 62258 unique words that occur more than once; other feature selection is not used. When tokenizing this data, we skip the UseNet headers (thereby discarding the subject line); tokens are formed from contiguous alphabetic characters, which are left unstemmed. The word counts of each document are scaled such that each document has constant length, with potentially fractional word counts. Our preliminary experiments with 20 Newsgroups indicated that naive Bayes classification was more accurate with this word count normalization.

The 20 Newsgroups data set was collected from UseNet postings over a period of several months in 1993. Naturally, the data have time dependencies—articles

[4] These data sets are available on the Internet at http://www.cs.cmu.edu/∼textlearning and http://www.research.att.com/∼lewis.

nearby in time are more likely to be from the same thread, and because of occasional quotations, may contain many of the same words. In practical use, a classifier for this data set would be asked to classify future articles after being trained on articles from the past. To preserve this scenario, we create a test set of 4000 documents by selecting by posting date the last 20% of the articles from each newsgroup. An unlabeled set is formed by randomly selecting 10000 documents from those remaining. Labeled training sets are formed by partitioning the remaining 6000 documents into non-overlapping sets. The sets are created with equal numbers of documents per class. For experiments with different labeled set sizes, we create up to ten sets per size; obviously, fewer sets are possible for experiments with labeled sets containing more than 600 documents. The use of each non-overlapping training set comprises a new trial of the given experiment.

The WebKB data set (Craven et al., 2000) contains 8145 web pages gathered from university computer science departments. The collection includes the entirety of four departments, and additionally, an assortment of pages from other universities. The pages are divided into seven categories: student, faculty, staff, course, project, department and other. In this thesis, we use the four most populous non-other categories: student, faculty, course and project—all together containing 4199 pages. The task is to classify a web page into the appropriate one of the four categories. For consistency with previous studies with this data set (Craven et al., 2000), when tokenizing the WebKB data numbers are converted into a time or a phone number token, if appropriate, or otherwise a sequence-of-length-n token.

We did not use stemming or a stoplist; we found that using a stoplist actually hurt performance. For example, the stopword *my* is an excellent indicator of a student homepage and is the fourth-ranked word by mutual information. We limit the vocabulary to the 300 most informative words, as measured by average mutual information with the class variable. This feature selection method is commonly used for text (Yang & Pedersen, 1997; Koller & Sahami, 1997; Joachims, 1997). We selected this vocabulary size by running leave-one-out cross-validation on the training data to optimize classification accuracy.

The WebKB data set was collected as part of an effort to create a crawler that explores previously unseen computer science departments and classifies web pages into a knowledge-base ontology. To mimic the crawler's intended use, and to avoid reporting performance based on idiosyncrasies particular to a single department, we test using a leave-one-university-out approach. That is, we create four test sets, each containing all the pages from one of the four complete computer science departments. For each test set, an unlabeled set of 2500 pages is formed by randomly selecting from the remaining web pages. Non-overlapping training sets are formed by the same method as in 20 Newsgroups.

The Reuters 21578 Distribution 1.0 data set consists of 12902 articles and 90 topic categories from the Reuters newswire. Following several other studies (Joachims, 1998; Liere & Tadepalli, 1997) we build binary classifiers for each of the ten most populous classes to identify the news topic. Since the documents in this data set can have multiple class labels, each category is traditionally evaluated with a binary classifier. We use all the words inside the <TEXT> tags, including the title and the dateline, except that we remove the REUTER and &# tags that occur at the top and bottom of every document. We use a stoplist, but do not stem.

In Reuters, the best vocabulary size differs depending on which category is of interest. This variance in optimal vocabulary size is unsurprising. As previously noted (Joachims, 1997), categories like wheat and corn are known for a strong correspondence between a small set of words (like their title words) and the categories, while categories like acq are known for more complex characteristics. The categories with narrow definitions attain best classification with small vocabularies, while those with a broader definition require a large vocabulary. The vocabulary size for each Reuters trial is selected by optimizing accuracy as measured by leave-one-out cross-validation on the labeled training set.

As with the 20 Newsgroups data set, there are time dependencies in Reuters. The standard 'ModApte' train/test split divides the articles by time, such that the later 3299 documents form the test set, and the earlier 9603 are available for training. In our experiments, 7000 documents from this training set are randomly selected to form the unlabeled set. From the remaining training documents, we randomly select up to ten non-overlapping training sets of just ten positively labeled documents and 40 negatively labeled documents, as previously described for the other two data sets. We use a non-uniform number of labelings across the classes because the negative class is much more frequent than the positive class in all of the binary Reuters classification tasks.

Results on Reuters are reported as both classification accuracy and precision-recall breakeven points, a standard information retrieval measure for binary classification.

Accuracy is not always a good performance metric here because very high accuracy can be achieved by always predicting the negative class. The task on this data set is less like classification than it is like filtering—find the few positive examples from a large sea of negative examples. Recall and precision capture the inherent duality of this task, and are defined as:

$$\text{Recall} = \frac{\text{\# of correct positive predictions}}{\text{\# of positive examples}} \tag{2.12}$$

$$\text{Precision} = \frac{\text{\# of correct positive predictions}}{\text{\# of positive predictions}}. \tag{2.13}$$

The classifier can achieve a trade-off between precision and recall by adjusting the decision boundary between the positive and negative class away from its previous default of $\mathrm{P}(c_j|d_i;\hat{\theta}) = 0.5$. The precision-recall breakeven point is defined as the precision and recall value at which the two are equal (e.g., Joachims, 1998).

The algorithm used for experiments with EM is described in Table 2.1. Classification results are reported as classification accuracy averages over all trials with the same number of labeled or unlabeled documents, as appropriate. When posterior model probability is reported and shown on graphs, some additive and multiplicative constants are dropped, but the relative values are maintained.

The computational complexity of EM is not prohibitive. Each iteration requires classifying the training documents (E-step), and building a new classifier (M-step). In our experiments, EM usually converges after about 10 iterations. The wall-clock time to read the document-word matrix from disk, build an EM model by iterating to convergence, and classify the test documents is less than one minute for the WebKB data set, and less than three minutes for 20 Newsgroups. The 20 Newsgroups data set takes longer because it has more documents and more words in the vocabulary.

2.6.2 EM with Unlabeled Data Increases Accuracy

We first consider the use of basic EM to incorporate information from unlabeled documents. Figure 2.2 shows the effect of using basic EM with unlabeled data on the 20 Newsgroups data set. The vertical axis indicates average classifier accuracy on test sets, and the horizontal axis indicates the amount of labeled training data on a log scale. We vary the amount of labeled training data, and compare the classification

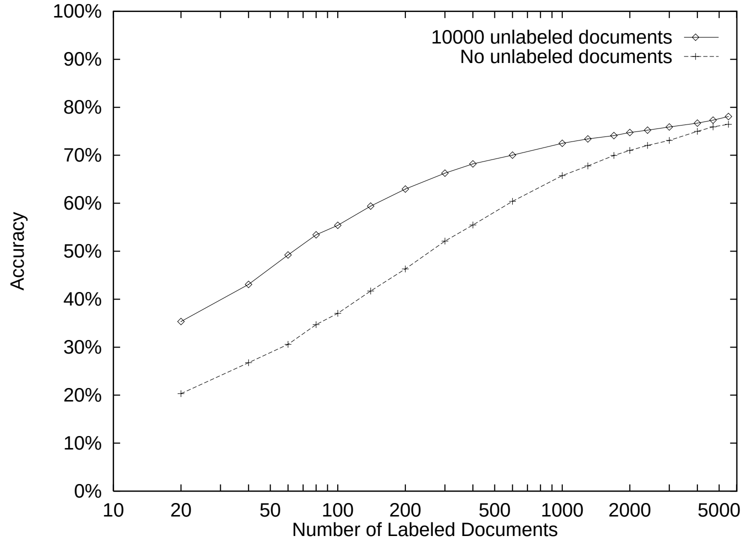

Figure 2.2: Classification accuracy on the 20 Newsgroups data set, both with and without 10,000 unlabeled documents. With small amounts of training data, using EM yields more accurate classifiers. With large amounts of labeled training data, accurate parameter estimates can be obtained without the use of unlabeled data, and the two methods begin to converge.

accuracy of traditional naive Bayes (no unlabeled documents) with an EM learner that has access to 10000 unlabeled documents.

EM performs significantly better than traditional naive Bayes. For example, with 300 labeled documents (15 documents per class), naive Bayes reaches 52% accuracy while EM achieves 66%. This represents a 30% reduction in classification error. Note that EM also performs well even with a very small number of labeled documents; with only 20 documents (a single labeled document per class), naive Bayes obtains 20%, EM 35%. As expected, when there is a lot of labeled data, and the naive Bayes learning curve is close to a plateau, having unlabeled data does not help nearly as much, because there is already enough labeled data to accurately estimate the classifier parameters. With 5500 labeled documents (275 per class), classification accuracy increases from 76% to 78%. Each of these results is statistically significant ($p < 0.05$).[5] Another way to view these results is to consider how unlabeled data reduce the need for labeled training examples. For example, to reach 70% classification accuracy, naive Bayes requires 2000 labeled examples, while EM requires only 600

[5] For all statistical results in this chapter, when the number of labeled examples is small, we have multiple trials, and use paired t-tests. When the number of labeled examples is large, we have a single trial, and report results instead with a McNemar test. These tests are discussed further by Dietterich (1998).

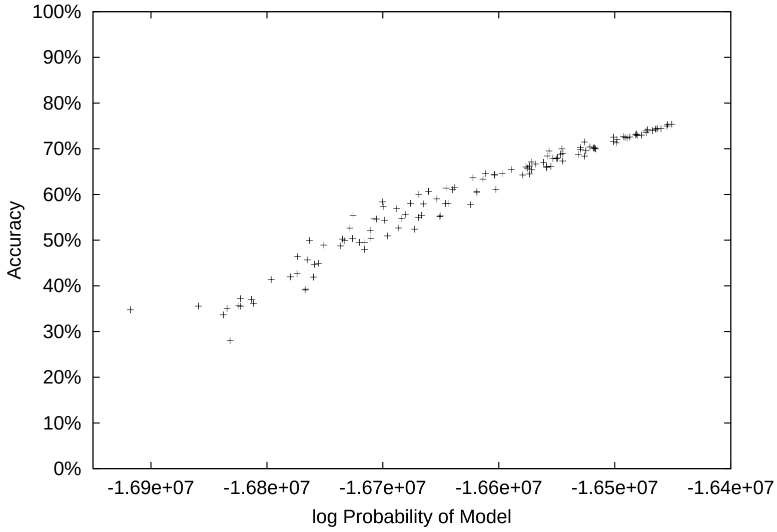

Figure 2.3: A scatterplot showing the correlation between the posterior model probability and the accuracy of a model trained with labeled and unlabeled data. The strong correlation implies that model probability is a good optimization criteria for the 20 Newsgroups dataset.

labeled (and 10000 unlabeled) examples to achieve the same accuracy. These results indicate that incorporating unlabeled data into supervised learning with generative models can significantly improve the accuracy of text classification, especially when labeled data are sparse.

How does EM find more accurate classifiers? It does so by optimizing on posterior model probability, not classification accuracy directly. If our generative model were perfect then we would expect model probability and accuracy to be correlated and EM to be helpful. But, we know that our simple generative model does not capture many of the properties contained in the text. Our 20 Newsgroups results show that we do not need a perfect model for EM to help text classification. Generative models are representative enough for the purposes of text classification if model probability and accuracy are correlated, allowing EM to indirectly optimize accuracy.

To illustrate this more definitively, let us look again at the 20 Newsgroups experiments, and empirically measure this correlation. Figure 2.3 demonstrates the correlation—each point in the scatterplot is one of the labeled and unlabeled splits from Figure 2.2. The labeled data here are used only for setting the EM initialization and are not used during iterations.[6] We plot classification performance as accuracy

[6] Section 4.2 shows that using the labeled data just for setting the starting point gives essentially the same performance when we also use it in the EM iterations. We exclude the labeled data from

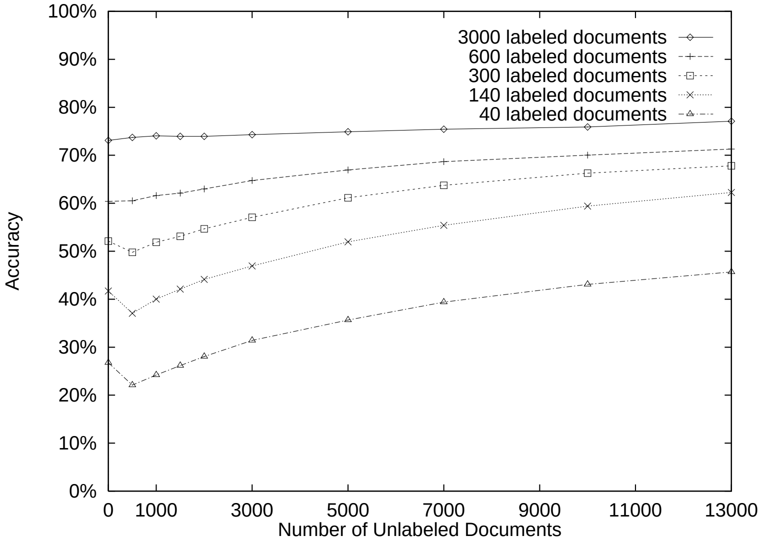

Figure 2.4: Classification accuracy while varying the number of unlabeled documents. The effect is shown on the 20 Newsgroups data set, with 5 different amounts of labeled documents, by varying the amount of unlabeled data on the horizontal axis. Having more unlabeled data helps. Note the dip in accuracy when a small amount of unlabeled data is added to a small amount of labeled data. We hypothesize that this is caused by extreme, almost 0 or 1, estimates of component membership, $\mathrm{P}(c_j|d_i,\hat{\theta})$, for the unlabeled documents (as caused by naive Bayes' word independence assumption).

on the test data and show the posterior model probability.

For this dataset, classification accuracy and model probability are in good correspondence. The correlation coefficient between accuracy and model probability is 0.9798, a very strong correlation indeed. We can take this as a post-hoc verification that this dataset is amenable to using unlabeled data via a generative model approach. The optimization criteria of model probability is applicable here because it is in tandem with accuracy.

We have shown that as the amount of labeled data increases, accuracy also increases. In Figure 2.4 we consider the effect of varying the amount of unlabeled data. For five different quantities of labeled documents, we hold the number of labeled documents constant, and vary the number of unlabeled documents in the horizontal axis. Naturally, having more unlabeled data helps, and it helps more when there is less labeled data.

Notice that adding a small amount of unlabeled data to a small amount of labeled data actually hurts performance. We hypothesize that this occurs because the word

the iterations to allow model probability numbers to be comparable across trials.

Iteration 0	Iteration 1	Iteration 2
intelligence	*DD*	*D*
DD	*D*	*DD*
artificial	lecture	lecture
understanding	cc	cc
*DD*w	$D^\star$	*DD:DD*
dist	*DD:DD*	due
identical	handout	$D^\star$
rus	due	homework
arrange	problem	assignment
games	set	handout
dartmouth	tay	set
natural	*DD*am	hw
cognitive	yurttas	exam
logic	homework	problem
proving	kfoury	*DD*am
prolog	sec	postscript
knowledge	postscript	solution
human	exam	quiz
representation	solution	chapter
field	assaf	ascii

Table 2.2: Lists of the words most predictive of the course class in the WebKB data set, as they change over iterations of EM for a specific trial. By the second iteration of EM, many common course-related words appear. The symbol D indicates an arbitrary digit.

independence assumption of naive Bayes leads to overly-confident $\mathrm{P}(c_j|d_i, \hat{\theta})$ estimates in the E-step, which cause each unlabeled document to be heavily weighted to only one class even without strong evidence for this. (Without this bias in naive Bayes, the E-step would spread the unlabeled data more evenly across the classes.) When the number of unlabeled documents is large, however, this problem disappears because the unlabeled set provides a large enough sample to smooth out the sharp discreteness of naive Bayes' overly-confident classification.

To provide some intuition about why EM works, we present a detailed trace of the evolution of the classifier over the course of several EM iterations. Table 2.2 shows the changing definition of the course class in the WebKB dataset. Each column shows the ordered list of words that the model indicates are most predictive of the course

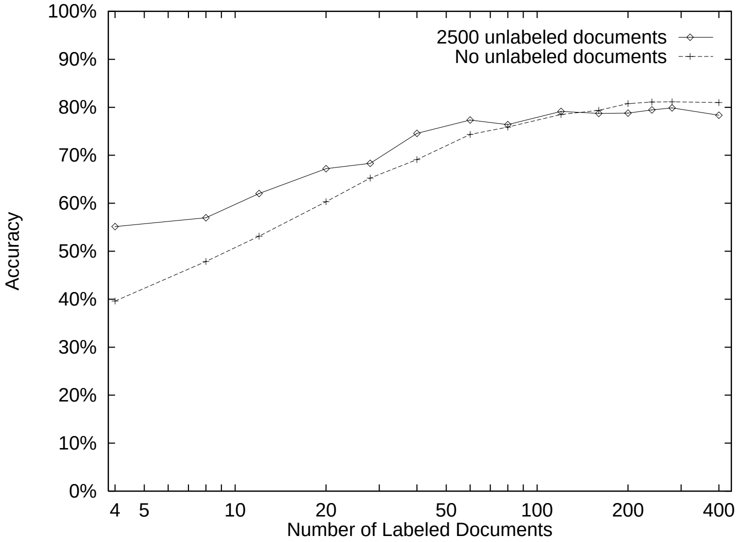

Figure 2.5: Classification accuracy on the WebKB data set, both with and without 2500 unlabeled documents. When there are small numbers of labeled documents, EM improves accuracy. When there are many labeled documents, however, EM degrades performance slightly—indicating a misfit between the data and the assumed generative model.

class. Words are judged to be predictive using a weighted log likelihood ratio.[7] The symbol D indicates an arbitrary digit. At Iteration 0, the parameters are estimated from a randomly-chosen single labeled document per class. Notice that the course document seems to be about a specific Artificial Intelligence course at Dartmouth. After two EM iterations with 2500 unlabeled documents, we see that EM has used the unlabeled data to find words that are more generally indicative of courses. The classifier corresponding to the first column achieves 50% accuracy; when EM converges, the classifier achieves 71% accuracy.

[7] The weighted log likelihood ratio used to rank the words in Figure 2.2 is:

$$\mathrm{P}(w_t|c_j;\hat{\theta}) \log \left(\frac{\mathrm{P}(w_t|c_j;\hat{\theta})}{\mathrm{P}(w_t|\neg c_j;\hat{\theta})} \right), \tag{2.14}$$

which can be understood in information-theoretic terms as word w_t's contribution to the average inefficiency of encoding words from class c_j using a code that is optimal for the distribution of words in $\neg c_j$. The sum of this quantity over all words is the Kullback-Leibler divergence between the distribution of words in c_j and the distribution of words in $\neg c_j$ (Cover & Thomas, 1991).

	Precision-Recall Breakeven			Classification Accuracy		
Category	NB	EM	NB vs. EM	NB	EM	NB vs. EM
acq	69.4	70.7	+3.3	86.9	81.3	-5.6
corn	44.3	44.6	+0.3	94.6	93.2	-1.4
crude	65.2	68.2	+3.0	94.3	94.9	+0.6
earn	91.1	89.2	-1.9	94.9	95.2	+0.3
grain	65.7	67.0	+1.3	94.1	93.6	-0.5
interest	44.4	36.8	-7.6	91.8	87.6	-4.2
money-fx	49.4	40.3	-9.1	93.0	90.4	-2.6
ship	44.3	34.1	-10.2	94.9	94.1	-0.8
trade	57.7	56.1	-1.6	91.8	90.2	-1.6
wheat	56.0	52.9	-3.1	94.0	94.5	+0.5

Table 2.3: Precision-recall breakeven and accuracy showing performance of binary classifiers on Reuters with naive Bayes (NB) and basic EM (EM) with labeled and unlabeled data.

2.6.3 EM with Unlabeled Data Can Hurt Accuracy

On some datasets, like 20 Newsgroups, EM increases accuracy. But with the WebKB data set, we see that the incorporation of unlabeled data can also decrease classification accuracy. The graph in Figure 2.5 shows the performance of basic EM (with 2500 unlabeled documents) on WebKB. Again, EM improves accuracy significantly when the amount of labeled data is small. When there are four labeled documents (one per class), traditional naive Bayes attains 40% accuracy, while EM reaches 55%. When there is a lot of labeled data, however, EM hurts performance slightly. With 240 labeled documents, naive Bayes obtains 81% accuracy, while EM does worse at 79%. Both of these differences in performance are statistically significant ($p < 0.05$), for three and two of the university test sets, respectively. Here EM hurts performance because the data do not fit the assumptions of the generative model—the mixture components that best explain the unlabeled data are not in precise correspondence with the class labels.

On WebKB EM hurts accuracy when there is a relatively large amount of labeled data. However, EM can also hurt performance when labeled data are sparse. Table 2.3 shows this for the Reuters dataset. When incorporating the unlabeled data into parameter estimation, both precision-recall breakeven and classification accuracy decrease more often than not. This indicates that our generative model is not accurate enough for this dataset. For each of the Reuters categories EM finds a significantly more probable model, given the evidence of the labeled and unlabeled data. But frequently this more probable model corresponds to a lower-accuracy classifier—not what we would hope for.

2.7 Discussion

From our experimental results, we see on two domains that when labeled data are sparse, using unlabeled data helps classification considerably. This is a significant finding because it demonstrates that some text classification tasks can be addressed with significantly less human labeling effort than before. It was not clear from the beginning whether maximizing the probability of our simple generative model was a reasonable optimization criteria when using unlabeled data. For some text classification tasks, likelihood and accuracy correspond to each other; the generative model approach works well in these cases.

However, on two domains we see unlabeled data hurting, sometimes with sparse labeled data and sometimes with plentiful labeled data. There are several reasonable hypotheses that could explain why sometimes EM does not do so well.

One hypothesis that would explain the mixed performance of EM goes back to the assumed generative model. As we discussed before, text documents are blatantly *not* generated by a mixture of multinomials process. EM maximizes the probability subject to the assumption that the generative model is correct, but there are no guarantees that EM will produce a reasonable classifier for our simple models of text. EM will give high-probability parameters based on the unlabeled data, but they may not give good accuracy.

Take, for example, the WebKB dataset. Even when labeled data were sufficient to saturate the learner, we still had an order of magnitude more unlabeled data. In this case, the great majority of the data determining the parameter estimates comes from the unlabeled set. We can think of EM as almost performing unsupervised clustering in a mixtures of multinomials world, since the model is mostly positioning the mixture components to maximize the likelihood of the unlabeled documents. When the mixture model assumptions are even just a little bit off, the natural clustering of the unlabeled data may produce mixture components that are not in correspondence with the class labels, and are therefore detrimental to classification accuracy. For other text tasks, posterior probability maximization has also been detrimental when the amount of labeled data is reasonable. While EM increased the likelihood of the parameters, the accuracy of part-of-speech taggers (Merialdo, 1994; Elworthy, 1994) and information extractors (McCallum et al., 2000) went down. For these tasks, each example is defined by just a small amount of local context. Here, the correctness of the model is much more important, because there are only a few, very correlated features. In text classification, documents are much longer, and not as sensitive to the local correctness of the generative model.

It is also understandable that the performance on Reuters often decreased with the addition of unlabeled data. Consider the validity of the generative model for this newswire domain. One multinomial component models documents about a single topic, like trade. Documents for all other topics are modeled by the second multinomial component. It is overly-optimistic to try to model "all documents but trade" with a single multinomial; the entire newswire has many sub-clusters of documents in it. Also, consider the relation between the desired classification task and the most probable clustering of the data with only two multinomials. It seems unlikely that the natural clustering with two multinomials would correspond to this unusual separation of the data. A better generative model would take into account both the classification task and the natural distribution of unlabeled documents.

A second hypothesis to explain why EM does poorly in using unlabeled data is that EM gets caught in poor local maxima in model probability space during the hill-climbing process. This trouble can arise even when model probability and accuracy are in good correspondence. EM is guaranteed to converge only to a saddlepoint or local maxima and does not guarantee the best global solution. In fact, with the high-dimensional parameter space used by our mixture of multinomials, it is a certainty that local maxima are everywhere. For example, in the case of a mixture of two Gaussians, each data point introduces a singularity into the likelihood surface (Day, 1969). In practice, statisticians have always been concerned about poor local maxima with EM. The most standard approach for combatting this problem is to run EM many times from randomly chosen starting points, and to select the local maximum with the highest probability. We have not adopted this approach here because the presence of the labeled data already indicate a good starting point. However, other approaches can be used to find more probable maxima that have not been addressed in this chapter. If indeed, models with higher probability could be found, these may correspond to classifiers with higher accuracy.

The remainder of this thesis explores these two hypotheses in detail. Chapter 3 addresses the concerns of the first hypothesis by changing the generative model in two different ways. These changes bring model probability and accuracy into correlation for complex datasets. Chapter 4 addresses the second hypothesis and shows two ways to reduce the cost incurred by local maxima. These approaches further increase the benefit of unlabeled data when the model is representative.

Chapter 3

Enhancing the Generative Model

When the generative modeling assumptions vary too much from reality the addition of unlabeled data hurts classification accuracy. This happens when models with high probability on the unlabeled data do not correspond to high-accuracy classifiers on test data. This negative effect can often be reversed by using a more accurate statistical model for the domain in question. There may be further structure between or within classes that is not represented in the generative model. For example, we may improve the model to represent sub-topic structure by modeling each class as a mixture of several subtopics, instead of just a single topic. Or, we may capture super-topic structure by integrating a model of hierarchical relationships between the classes. Experimental evidence shows that with these more sophisticated model classes, higher-probability models give more accurate text classifiers. When the use of the basic model hurts classification performance, these improved models allow supervised learning to benefit by the inclusion of unlabeled data.

3.1 Introduction

In the previous chapter, we assumed that a simple statistical model had generated the text documents in our dataset. By taking a model posterior maximization approach we were able to estimate parameters of the model from all the data—both the labeled and the unlabeled. The addition of the unlabeled data enabled us to find much more probable parameters with EM than we would using labeled data alone—especially when labeled data were sparse. We saw that in some domains the probability of

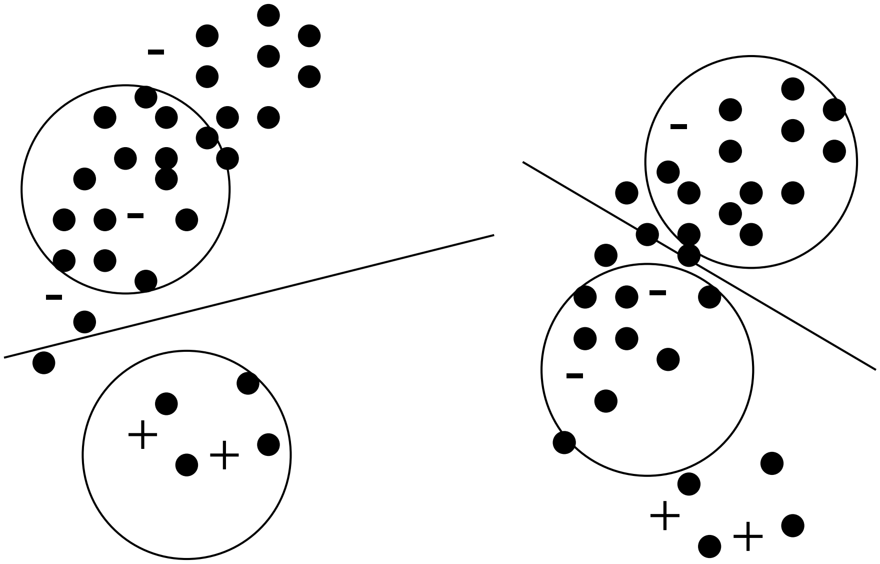

Figure 3.1: An example of learning with labeled and unlabeled data when the generative model is not representative. Here, the presence of the unlabeled data hurts classification accuracy.

the model parameters was strongly correlated with classification accuracy; here using the unlabeled data to select parameterizations gave more accurate classifiers. In other domains model probability and accuracy are not well-correlated because the generative model was not representative; on these datasets using unlabeled data hurt classification performance. In this chapter we confront this problem by changing the generative assumptions in two different ways to make our models representative enough that unlabeled data will help text classification on these more challenging domains.

Why will unrepresentative models decouple the relationship between accuracy and model probability? Consider a binary classification task on the non-text dataset shown in Figure 3.1. If we assume a generative model that is a mixture of two Gaussians, one per class, with fixed uniform covariance matrices, we can estimate its parameters from the data shown. Note this is the wrong generative model, as the two clusters in the data would best be represented with quite different covariance matrices. Yet, if we use maximum a posteriori methods to build a classifier (indicated by the linear separator) using only the five labeled datapoints shown, it performs reasonably accurately. Consider what happens if we add in all the unlabeled data, indicated by the dots. Now, when model maximization is performed, the large mass of negative examples not captured by the small positive Gaussian will draw over the positive

Gaussian during the iterations of EM. This will result in the scenario shown on the right hand side, where classification accuracy is poor indeed.

What went wrong? Our generative model was not representative of the data. EM found that the most probable parameterization put each mixture covering one side of the negative class, leaving the positive class unattended. If the model had been more representative and had not fixed the covariance matrices unrealistically, EM would be able to model the data better and give an accurate classifier.

For our text classification domains we have made three strong assumptions about the generative model of text documents. They are (1) the documents are generated by a mixture model, (2) there is a one-to-one correspondence between mixture components and classes, and (3) each mixture component is a multinomial distribution over words. As authors, people do not actually compose documents in this fashion. Yet for some domains posterior model probability and accuracy are correlated for this generative model, allowing unlabeled data to be helpful for classification. In cases where this correlation does not hold, our toy example suggests that we should modify our assumptions to be more representative of the true data distribution. In this chapter we relax in turn each of the first two assumptions about text documents.

In Section 3.2 we consider the assumption that each class is modeled by a single mixture component. Often a classification task will demand that a single class cover a complicated and multi-faceted topic. In this case it would be unrealistic to model this class with only one mixture component. A more representative model would instead use multiple mixture components to allow each sub-topic of the class to be expressed separately. In the previous chapter the Reuters dataset had this characteristic, and with the original generative model, unlabeled data hurt classification performance. With multiple mixture components per class, including unlabeled data into parameter estimation makes our text classifiers more accurate.

In Section 3.3 we consider the assumption that the data are produced by a mixture model. This means that the documents of each class are unrelated to the others. However, it is often the case that there is a natural hierarchical organization of the classes, with some classes being more similar than others. Given such a hierarchy, we model the class relationships to allow parameters shared between classes to be estimated more accurately. The Cora dataset, a collection of computer science research papers, has a hierarchy of the sub-fields of CS. Experimental results show that by leveraging the hierarchy we can improve classification accuracy using unlabeled data,

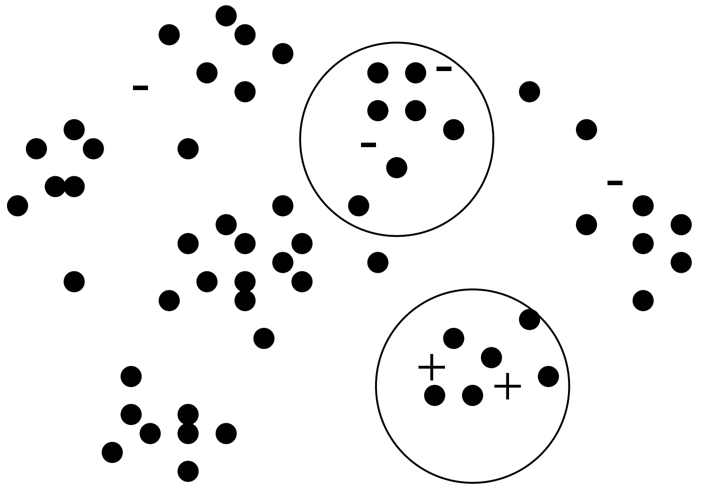

Figure 3.2: An example of data that violates the assumption of a one-to-one correspondence between mixture components and Gaussians. With the labeled and unlabeled data indicated, maximizing the probability of an incorrect model gives a poor classifier.

where with the basic model classification accuracy decreases.

3.2 Modeling Sub-topic Structure

The second assumption of our generative model states that there is a one-to-one correspondence between classes and components in the mixture model. When this assumption is strongly violated, incorporating unlabeled data with model estimation can hurt classification performance. For example, let us return to our Gaussian mixture model of the previous section. There, we saw that the assumption that all mixture components had the same covariance matrix was unrealistic and harmful. Figure 3.2 shows an example of some labeled and unlabeled data that violates instead the assumption that each class has exactly one mixture component. The positive class is well-modeled with one Gaussian, but the negative class is not. The data are distributed like a "clumpy sea of examples" with the positive class being one small island.

What would be a more representative model? Instead of modeling a sea of negative examples with a single mixture component, it might be better to model it with many components. In this way, each negative component could, after maximization, capture one clump of the sea of examples. Figure 3.3 shows the same data modeled with six mixture components for the negative class. Here, we see a much more realistic

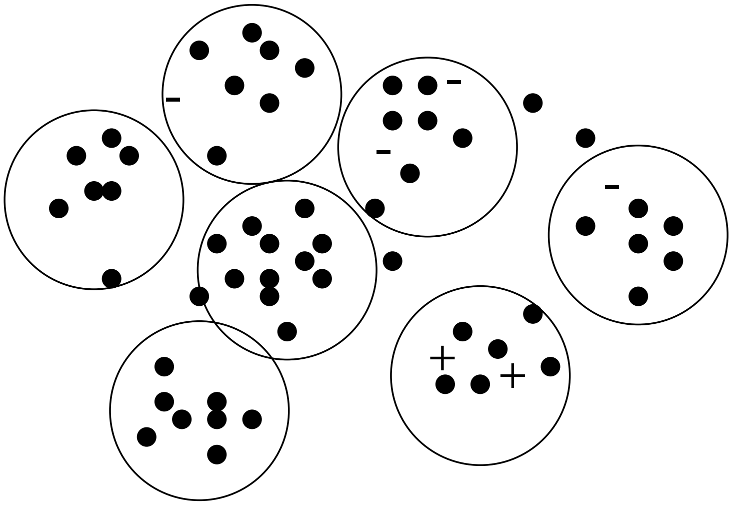

Figure 3.3: The same data as Figure 3.2, but this time modeled with six mixture components for the negative class. Here the assumed generative model is much more representative.

modeling of the data.

This section takes exactly the approach suggested by this example for text data, and relaxes the assumption of a one-to-one correspondence between mixture components and classes. We replace it with a less restrictive assumption: a *many*-to-one correspondence between mixture components and classes. This allows us to model the sub-topic structure of a class.

In text classification, an analogous scenario is not unusual. Consider the task of text filtering, where we want to identify a small well-defined class of documents from a very large pool or stream of documents. One example might be a system that watches a network administrator's incoming emails to identify the rare emergency situation that would require paging her on vacation. Modeling the non-emergency emails as the negative class with only one multinomial distribution will result in an unrepresentative model. The negative class contains emails with a variety of sub-topics: personal emails, non-emergency requests, spams, and many more. Perhaps with multiple mixture components we could more accurately model the negative class. This would allow model probability maximization with labeled and unlabeled data to give a classifier that provides significantly higher accuracy.

In Section 3.2.1 we show how to relax our assumptions and how to use EM for the new generative model to estimate highly probable parameters given our data. In Section 3.2.2 we return to the problematic Reuters dataset where incorporating unlabeled data with the original model hurt performance. Section 3.2.3 shows that with this more representative model, improving the model probability with EM also improves classification accuracy.

3.2.1 EM with Multiple Mixture Components per Class

With a many-to-one correspondence between mixture components and classes, the original generative model of the previous chapter is no longer applicable. The new generative model must account for the sub-topic structure. As in the old model, we first pick a class with a biased die roll. Each class has several sub-topics; we next pick one of these sub-topics, again with a biased die roll. Now that the sub-topic is determined, the document's words are generated. We do this by first picking a length (independently of sub-topic and class) and then draw the words from the sub-topic's multinomial distribution.

Unlike previously, there are now two missing values for each unlabeled document—its class and its sub-topic. Even for the labeled data there are missing values; although the class is known, its sub-topic is not. Since we do not have access to these missing class and sub-topic labels, we must use a technique such as EM to estimate local MAP generative parameters. As in Section 2.5, EM is instantiated as an iterative algorithm that alternates between estimating the values of missing class and sub-topic labels, and calculating the MAP parameters using the estimated labels. After EM converges to high-probability parameter estimates the generative model can be used for text classification by turning it around with Bayes' rule. Table 3.1 gives an overview of the modified EM algorithm.

To derive the details of this algorithm we use as a starting point the notation introduced in Chapter 2. The new generative model specifies a separation between mixture components and classes. Instead of using c_j to denote both of these, $c_j \in \mathcal{C}$ now denotes only the jth mixture component. We write $t_a \in \mathcal{T}$ for the ath class ("topic"); when component c_j belongs to class t_a, then $q_{aj} = 1$, and otherwise 0. This represents the pre-determined, deterministic, many-to-one mapping between mixture components and classes. We indicate the class label and sub-topic label of a document by x_i and y_i, respectively. Thus if document d_i was generated by mixture component c_j we say $y_i = c_j$, and if the document belongs to class t_a then we say $x_i = t_a$.

If all the class and sub-topic labels were known for our dataset, finding MAP

- **Inputs:** Collections $\mathcal{D}^l$ of labeled documents and $\mathcal{D}^u$ of unlabeled documents.
- Set the number of mixture components per class by cross-validation (see Sections 3.2.3).
- Initialize each mixture component by randomly assigning non-zero $\mathrm{P}(c_j|d_i;\hat{\theta})$ for each labeled document based on the document's class label.
- Build an initial classifier $\hat{\theta}$ from the labeled documents only. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \arg\max_\theta \mathrm{P}(\mathcal{D}|\theta)\mathrm{P}(\theta)$ (see Equations 3.1, 3.2 and 3.3).
- Loop while classifier parameters improve, measured by $\Delta l_c(\theta|\mathcal{D};\mathbf{z})$, the change in complete log probability of generative model:
    - **(E-step)** Use the current classifier, $\hat{\theta}$, to estimate the class and sub-topic memberships of each document (see Equations 3.4 and 3.5). Restrict the membership probability estimates of labeled documents to be zero for components associated with other classes and renormalize.
    - **(M-step)** Re-estimate the classifier, $\hat{\theta}$, given the estimated component membership of each document. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \arg\max_\theta \mathrm{P}(\mathcal{D}|\theta)\mathrm{P}(\theta)$ (see Equations 3.1, 3.2 and 3.3).
- **Output:** A classifier, $\hat{\theta}$, that takes an unlabeled document and predicts a class label using Equation 3.5.

Table 3.1: The modified algorithm for integrating unlabeled data with EM when using multiple mixture components per class.

estimates for the generative parameters would a straightforward application of closed-form equations similar to those for naive Bayes seen in Section 2.4. The formula for the word probability parameters is identical to Equation 2.6 for naive Bayes:

$$\hat{\theta}_{w_t|c_j} \equiv \mathrm{P}(w_t|c_j;\hat{\theta}) = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} N(w_t,d_i)\mathrm{P}(y_i = c_j|d_i)}{|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|\mathcal{D}|} N(w_s,d_i)\mathrm{P}(y_i = c_j|d_i)}. \tag{3.1}$$

The class probabilities are analogous to Equation 2.7, but using the new notation for classes instead of components:

$$\hat{\theta}_{t_a} \equiv \mathrm{P}(t_a|\hat{\theta}) = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} \mathrm{P}(x_i = t_a|d_i)}{|\mathcal{T}| + |\mathcal{D}|}. \tag{3.2}$$

The sub-topic probabilities are similar, except they are estimated only with reference to other documents in that component's class:

$$\hat{\theta}_{c_j} \equiv \mathrm{P}(c_j|t_a;\hat{\theta}) = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} \mathrm{P}(y_i = c_j|d_i)}{\sum_{j=1}^{|\mathcal{C}|} q_{aj} + \sum_{i=1}^{|\mathcal{D}|} \mathrm{P}(x_i = t_a|d_i)}. \tag{3.3}$$

At classification time, we must estimate class membership probabilities for an unlabeled document. This is done by first calculating sub-topic membership and then summing over sub-topics to get overall class probabilities. Sub-topic membership is calculated analogously to mixture component membership for naive Bayes, with a small adjustment to account for the presence of two priors (class and sub-topic) instead of just one:

$$\mathrm{P}(y_i = c_j|d_i;\hat{\theta}) = \frac{\sum_{a=1}^{|\mathcal{T}|} q_{aj}\mathrm{P}(t_a|\hat{\theta})\mathrm{P}(c_j|t_a;\hat{\theta}) \prod_{k=1}^{|d_i|} \mathrm{P}(w_{d_{i,k}}|c_j;\hat{\theta})}{\sum_{r=1}^{|\mathcal{C}|} \sum_{b=1}^{|\mathcal{T}|} q_{br}\mathrm{P}(t_b|\hat{\theta})\mathrm{P}(c_r|t_b;\hat{\theta}) \prod_{k=1}^{|d_i|} \mathrm{P}(w_{d_{i,k}}|c_r;\hat{\theta})}. \tag{3.4}$$

Overall class membership is calculated with a sum of probability over all of the class's sub-topics:

$$\mathrm{P}(x_i = t_a|d_i;\hat{\theta}) = \sum_{j=1}^{|\mathcal{C}|} q_{aj}\mathrm{P}(y_i = c_j|d_i;\hat{\theta}) \tag{3.5}$$

These equations for supervised learning are applicable only when all the training documents have both class and sub-topic labels. Without these we use EM. The derivation of the EM process used to find parameters of this model from labeled and unlabeled data runs analogously to Section 2.5.1. The M-step, as with basic EM, builds maximum a posteriori parameter estimates for the multinomials and priors. This is done with Equations 3.1, 3.2, and 3.3, using the probabilistic class and sub-topic memberships estimated in the previous E-step.

In the E-step, for the unlabeled documents we calculate probabilistically-weighted sub-topic and class memberships (Equations 3.4 and 3.5). For labeled documents, we must estimate sub-topic membership. But, we know from its given class label that many of the sub-topic memberships must be zero—those sub-topics that belong to other classes. Thus we calculate sub-topic memberships as for the unlabeled data, but setting the appropriate ones to zero, and normalizing the non-zero ones over only those topics that belong to its class.

For the original generative model we initialized EM using the labeled data. We use the same approach for multiple mixture components, but there are no initial sub-topic labels provided. Thus, we randomly spread each labeled training example across the mixture components that belong to its class. That is, components are initialized by performing a randomized E-step in which $\mathrm{P}(c_j|d_i;\hat{\theta})$ is sampled from a uniform distribution over mixture components belonging to the document's class.

If we are given a set of class-labeled data, and a set of unlabeled data, we can now apply EM if there is some specification of the number of sub-topics for each class. However, this information is not typically available. As a result we must resort to some techniques for model selection. There are many commonly-used approaches to model selection such as cross validation, AIC, BIC and others. Since we do have the availability of a limited number of labeled documents, we use cross-validation to select the number of sub-topics for classification performance.

There is tension in this model selection process between complexity of the model and data sparsity. With as many sub-topics as there are documents, we can perfectly model the training data—each sub-topic covers one training document. With still a large number of sub-topics, we can accurately model existing data, but generalization performance will be poor. This is because each multinomial will have its many parameters estimated from only a few documents and will suffer from sparse data. With very few sub-topics, the opposite problem will arise. We will very accurately estimate the multinomials, but the model will be overly restrictive, and not representative of the true document distribution. Cross-validation should help in selecting a good compromise between these tensions with specific regard to classification performance.

Note that our use of multiple mixture components per class allows us to capture some dependencies between words on the class-level. For example, consider a sports class consisting of documents about both hockey and baseball. In these documents, the words *ice* and *puck* are likely to co-occur, and the words *bat* and *base* are likely to co-occur. However, these dependencies cannot be captured by a single multinomial distribution over words in the sports class. With multiple mixture components per class, one multinomial can cover the hockey sub-topic, and another the baseball sub-topic. In the hockey sub-topic, the word probability for *ice* and *puck* will be significantly higher than they would be for the whole class. This makes their co-occurrence more likely in hockey documents than it would be under a single multinomial assumption.

3.2.2 Dataset and Protocol

As a test domain for our new generative model we return to Reuters, the collection of newswire articles used in the previous chapter. Since the documents in this dataset can have multiple class labels, each category is traditionally evaluated with a binary classifier. Using the original generative model, the negative class covers up to 89 distinct categories, and we expect this task to strongly violate the assumption that all the data for the negative class are generated by a single mixture component. For this reason, we model the positive class with a single mixture component and the negative class with between one and forty mixture components, both with and without unlabeled data.

We pre-process the data and follow the experimental protocol as described in Section 2.6.1. Only ten positive and 40 negative documents are labeled, and 7000 are unlabeled. Classification results are reported both as precision-recall breakeven points and as classification accuracy. The algorithm used for experiments with EM is described in Table 3.1.

When leave-one-out cross-validation is performed in conjunction with EM, we make one simplification for computational efficiency. We first run EM to convergence with all the training data, and then subtract the word counts of each labeled document in turn before testing that document. Thus, when performing cross-validation for a specific combination of parameter settings, only one run of EM is required instead of one run of EM per labeled example. Note, however, that there are still some residual effects of the held-out document.

3.2.3 Experimental Results

Table 3.2 contains a summary of the precision-recall breakeven results on Reuters. The NB1 and EM1 columns reproduce the results from Section 2.6.3 with the original generative model of one mixture component per class. Remember that more often than not, incorporating unlabeled data with EM hurts performance. We hypothesize that because the negative class is truly multi-modal, fitting a single naive Bayes class with EM to the data does not accurately capture the negative class word distribution.

The NB* column shows the results of modeling the negative class with multiple mixture components, but with just the labeled data. In the NB* column, the number of components has been selected to optimize the best precision-recall breakeven point.

Category	NB1	NB*	EM1	EM*	EM* vs NB1	EM* vs NB*
acq	69.4	74.3 (4)	70.7	83.9 (10)	+14.5	+9.6
corn	44.3	47.8 (3)	44.6	52.8 (5)	+8.5	+5.0
crude	65.2	68.3 (2)	68.2	75.4 (8)	+10.2	+7.1
earn	91.1	91.6 (1)	89.2	89.2 (1)	-1.9	-2.4
grain	65.7	66.6 (2)	67.0	72.3 (8)	+6.3	+5.7
interest	44.4	54.9 (5)	36.8	52.3 (5)	+7.9	-2.6
money-fx	49.4	55.3 (15)	40.3	56.9 (10)	+7.5	+1.6
ship	44.3	51.2 (4)	34.1	52.5 (7)	+8.2	+1.3
trade	57.7	61.3 (3)	56.1	61.8 (3)	+4.1	+0.5
wheat	56.0	67.4 (10)	52.9	67.8 (10)	+11.8	+0.4

Table 3.2: Precision-recall breakeven points showing performance of binary classifiers on Reuters with traditional naive Bayes (NB1), multiple mixture components using just labeled data (NB*), basic EM (EM1) with labeled and unlabeled data, and multiple mixture components EM with labeled and unlabeled data (EM*). For NB* and EM*, the number of components is selected optimally for each trial, and the median number of components across the trials used for the negative class is shown in parentheses. Note that the multi-component model is more natural for Reuters, where the negative class consists of many topics. Using both unlabeled data and multiple mixture components per class increases performance over either alone, and over naive Bayes.

The median number of components selected across trials is indicated in parentheses beside the breakeven point. Note that even without unlabeled data, using this more complex representation improves performance over traditional naive Bayes. EM automatically finds a high-probability division of the negative class into sub-topics that help improve classification.

The column labeled EM* shows results of EM with multiple mixture components using labeled and unlabeled data, again selecting the best number of components. Here performance is better than both NB1 (traditional naive Bayes) and NB* (naive Bayes with multiple mixture components per class), where with only one component per class EM was worse. This increase with unlabeled data, measured over all trials of Reuters, is statistically significant ($p < 0.05$). This indicates that while the use of multiple mixture components increases performance over traditional naive Bayes, the combination of unlabeled data and multiple mixture components increases performance even more. By using a generative model that is more representative of the multi-modal data, increasing model likelihood with EM also increases classification

Category	NB1	NB*	EM1	EM*	EM* vs NB1	EM* vs NB*
acq	86.9	88.0 (4)	81.3	93.1 (10)	+6.2	+5.1
corn	94.6	96.0 (10)	93.2	97.2 (40)	+2.6	+1.2
crude	94.3	95.7 (13)	94.9	96.3 (10)	+2.0	+0.6
earn	94.9	95.9 (5)	95.2	95.7 (10)	+0.8	-0.2
grain	94.1	96.2 (3)	93.6	96.9 (20)	+2.8	+0.7
interest	91.8	95.3 (5)	87.6	95.8 (10)	+4.0	+0.5
money-fx	93.0	94.1 (5)	90.4	95.0 (15)	+2.0	+0.9
ship	94.9	96.3 (3)	94.1	95.9 (3)	+1.0	-0.4
trade	91.8	94.3 (5)	90.2	95.0 (20)	+3.2	+0.7
wheat	94.0	96.2 (4)	94.5	97.8 (40)	+3.8	+1.6

Table 3.3: Classification accuracy on Reuters with traditional naive Bayes (NB1), multiple mixture components using just labeled data (NB*), basic EM (EM1) with labeled and unlabeled data, and multiple mixture components EM with labeled and unlabeled data (EM*), as in Table 3.2.

accuracy.

Table 3.3 shows the same results as Table 3.2, but for classification accuracy, and not precision-recall breakeven. The general trends for accuracy are the same as for precision-recall. However for accuracy, the optimal number of mixture components for the negative class is greater than for precision-recall. By its nature precision-recall focuses on modeling the positive class, where accuracy focuses more on modeling the negative class, because it is much more frequent. By allowing more mixture components for the negative class, a more accurate model is achieved.

It is interesting to note that on average EM* uses more mixture components than NB*. This suggests that the addition of unlabeled data supports the use of a more complex model. Without the unlabeled examples, the data sparsity problems are more severe, and the best tradeoff between model representation and parameter generally lies with fewer mixture components per class. When unlabeled data are present and plentiful, more complex models are more representative and accurate.

We can provide scatterplots of model probability versus accuracy that demonstrate that they become correlated with multiple mixture components. We do so for the acq classification task, and compare the use of ten mixture components (the median number picked for acq) to the use of only one. To provide fair comparisons across

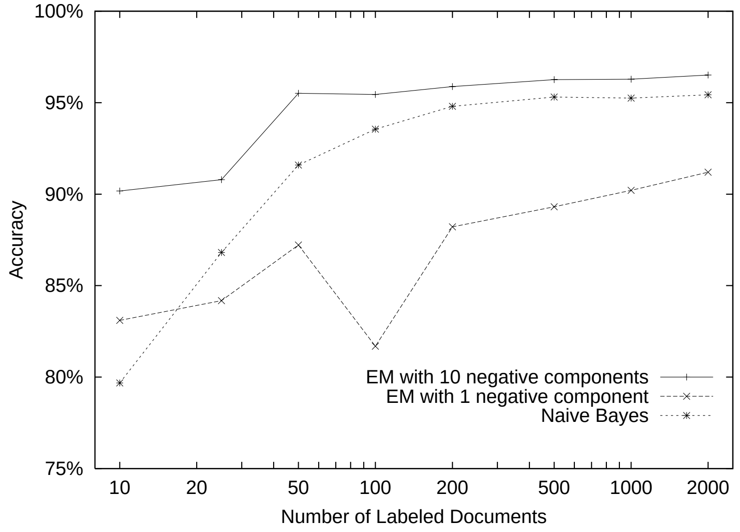

Figure 3.4: A comparison of models when using unlabeled data for the Reuters acq task. Accuracy is best when using ten mixture components for the negative class, and incorporating unlabeled data. With only one mixture component, EM finds high-likelihood models that do not correlate with high-accuracy classifiers.

different training set splits, we use the labeled data only to set the EM initialization[1] and fix the vocabulary to the top 500 words using all the training data (500 is the median vocabulary size selected). We vary the number of labeled data, using ten random training set splits for each amount. Figure 3.4 shows that using ten mixture components consistently gives the best classification accuracy compared to EM with a single mixture component or naive Bayes. Note that EM with just one component does quite poorly—significantly worse than naive Bayes.

The scatterplots tell why a single component is a poor representation. Figure 3.5 shows the correlation between model probability and classification accuracy for one mixture component (on the left) and ten mixture components (on the right). Note that with one component, the correlation is very strong ($r = -0.9906$), but in the wrong direction! Models with higher probability have significantly lower classification accuracy. By examining the solutions found by EM, we find that with only two mixture components (one positive, one negative) the most probable clustering of the data has one component that has the majority of negative documents and the second with most of the positive documents, but significantly more negative documents. Thus,

[1] Section 4.2 shows that using the labeled data just for setting the starting point gives essentially the same performance when we also use it in the EM iterations.

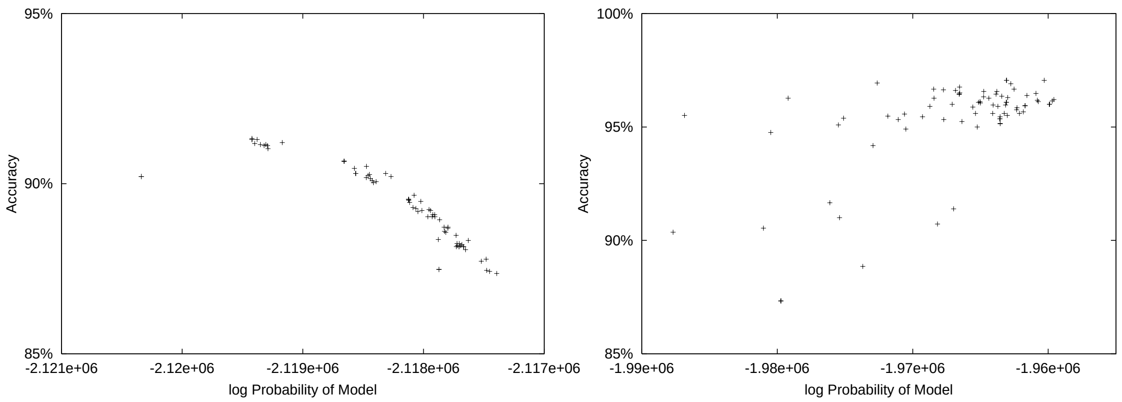

Figure 3.5: Scatterplots showing the relationship between model probability and classification accuracy for the Reuters acq task. On the left, with only one mixture component for the negative class, probability and accuracy are inversely proportional, exactly what we would not want. On the right, with ten mixture components for negative, there is a moderate positive correlation between model probability and classification accuracy.

the classes do not separate with high-probability models.

With ten mixture components the story is quite different. Figure 3.5 on the right shows a moderate correlation between model probability and classification accuracy in the right direction ($r = 0.5474$). For the solutions here, one component covers nearly all the positive documents and some, but not many, negatives. The other ten components are distributed through the remaining negative documents. This model is more representative of the data for our classification task because classification accuracy and model probability are correlated. This allows the beneficial use of unlabeled data through the generative model approach.

Tables 3.4 and 3.5 show the complete results for experiments using multiple mixture components with and without unlabeled data, respectively. Note that in general, using too many or too few mixture components hurts performance. With too few components, our assumptions are overly restrictive and our model is not representative. With too many components, there are more parameters to estimate from the same amount of data and we suffer from (unlabeled) data sparsity. With substantially larger amounts of unlabeled data, we hypothesize we would be able to support more mixture components, up to the point where the model is no longer representative.

One obvious question is how to automatically select the best number of mixture components without having access to the test set labels. We use leave-one-out cross-

Category	EM1	EM3	EM5	EM10	EM20	EM40
acq	70.7	75.0	72.5	77.1	68.7	57.5
corn	44.6	45.3	45.3	46.7	41.8	19.1
crude	68.2	72.1	70.9	71.6	64.2	44.0
earn	89.2	88.3	88.5	86.5	87.4	87.2
grain	67.0	68.8	70.3	68.0	58.5	41.3
interest	36.8	43.5	47.1	49.9	34.8	25.8
money-fx	40.3	48.4	53.4	54.3	51.4	40.1
ship	34.1	41.5	42.3	36.1	21.0	5.4
trade	56.1	54.4	55.8	53.4	35.8	27.5
wheat	52.9	56.0	55.5	60.8	60.8	43.4

Table 3.4: Performance of EM using different numbers of mixture components for the negative class and 7000 unlabeled documents. Precision-recall breakeven points are shown for experiments using between one and forty mixture components. Note that using too few or too many mixture components results in poor performance.

validation, with the computational short-cut that entails running EM only once (as described at the end of Section 3.2.2). Results from this technique (EM*CV), compared to naive Bayes (NB1) and the best EM (EM*), are shown in Table 3.6. Note that cross-validation does not perfectly select the number of components that perform best on the test set.

3.2.4 Discussion

In these experiments we see that it can be advantageous to explicitly model sub-topic structure for text classification when a single multinomial is too restrictive a model for a class. We do this by allowing multiple multinomial mixture components per class, and fit their parameters with EM. The use of the new model and the unlabeled data improves classification accuracy over naive Bayes with labeled data in all ten Reuters classification tasks.

When choosing how complex a model to use, our method of cross-validation consistently selects too few mixture components. By using cross-validation with the computational short-cut, we bias the model towards the held-out document, which, we hypothesize, favors the use of fewer components. The computationally expensive,

Category	NB1	NB3	NB5	NB10	NB20	NB40
acq	69.4	69.4	65.8	68.0	64.6	68.8
corn	44.3	44.3	46.0	41.8	41.1	38.9
crude	65.2	60.2	63.1	64.4	65.8	61.8
earn	91.1	90.9	90.5	90.5	90.5	90.4
grain	65.7	63.9	56.7	60.3	56.2	57.5
interest	44.4	48.8	52.6	48.9	47.2	47.6
money-fx	49.4	48.1	47.5	47.1	48.8	50.4
ship	44.3	42.7	47.1	46.0	43.6	45.6
trade	57.7	57.5	51.9	53.2	52.3	58.1
wheat	56.0	59.7	55.7	65.0	63.2	56.0

Table 3.5: Performance of EM using different numbers of mixture components for the negative class, but with no unlabeled data. Precision-recall breakeven points are shown for experiments using between one and forty mixture components.

Category	NB1	EM*	EM*CV	EM*CV vs NB1
acq	69.4	83.9 (10)	75.6 (1)	+6.2
corn	44.3	52.8 (5)	47.1 (3)	+2.8
crude	65.2	75.4 (8)	68.3 (1)	+3.1
earn	91.1	89.2 (1)	87.1 (1)	-4.0
grain	65.7	72.3 (8)	67.2 (1)	+1.5
interest	44.4	52.3 (5)	42.6 (3)	-1.8
money-fx	49.4	56.9 (10)	47.4 (2)	-2.0
ship	44.3	52.5 (7)	41.3 (2)	-3.0
trade	57.7	61.8 (3)	57.3 (1)	-0.4
wheat	56.0	67.8 (10)	56.9 (1)	+0.9

Table 3.6: Performance of using multiple mixture components when the number of components is selected via cross-validation (EM*CV) compared to optimal selection (EM*) and straight naive Bayes (NB1). Note that cross-validation usually selects too few components.

but complete, cross-validation should perform better. Other model selection methods may also perform better, while maintaining computational efficiency. These include more robust methods of cross-validation Ng (1997), Minimum Description Length (Rissanen, 1983), and Schuurman's (1997) metric-based approach, which also uses unlabeled data. Research on improved methods of model selection for our algorithm is an area of future work.

We believe that when learning with a generative model approach, situations with labeled and unlabeled data require a closer match between the data and the model than those using labeled data alone. If the intended target concept and model differ from the actual distribution of the data too strongly, then the use of unlabeled data will hurt instead of help performance. However, with just labeled data things are somewhat more resilient. For example, in Figure 3.1 we assumed equivalent covariance matrices for both classes, but with labeled data alone, the derived classifier is rather accurate. Similarly, naive Bayes for text or non-text can give good classification even when class probability estimates are poor, because only the linear boundary is important for classification (Domingos & Pazzani, 1997; Friedman, 1997). With unlabeled data, the dependence upon the generative model is much stronger, because the model is used to give the unlabeled data their estimated class labels. For the Reuters example, we did see improvements in the labeled data case with the expanded generative models. But with labeled and unlabeled data, the differences were much larger.

3.3 Modeling Super-topic Structure

In the previous section we modeled sub-topic structure by relaxing the assumption about the relationships between a class and mixture components. Now we model super-topic structure by changing the assumption about the relationship among classes themselves. Mixture models require that the parametric form of each class be independent of all other classes. In this section we model dependencies between the classes with hierarchical relationships.

In many text domains the classes of interest are arranged in a hierarchy. For example, there are many hierarchical organizations of the Web; Yahoo is the canonical example. The Dewey Decimal system assigns books in a hierarchical fashion. All patents are classified into a very large hierarchy of different fields of innovation. Research is also arranged hierarchically; the Computing Research Repository (*http://arxiv.org/*) uses the ACM Computer Science Hierarchy to categorize each submitted research paper.

With complex datasets with many classes even a large amount of unlabeled data

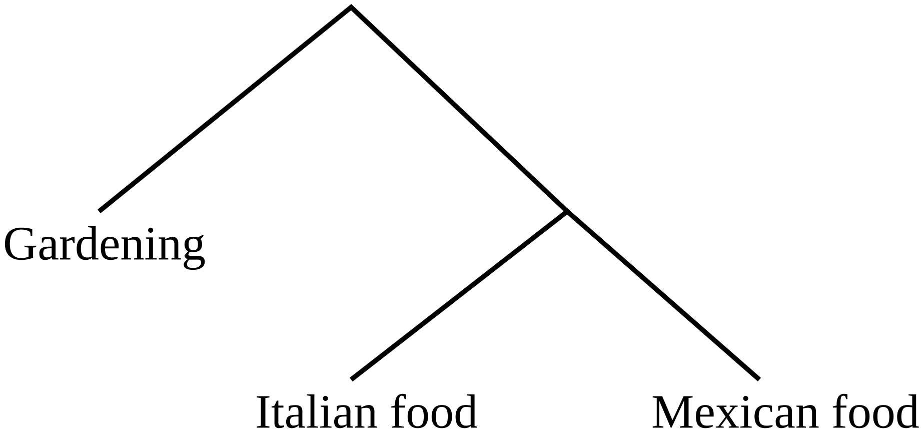

Figure 3.6: A simple hierarchy demonstrating relationships between classes.

may not be sufficient to model the document distribution accurately. When this happens, likelihood maximization will find parameters that model the unlabeled data well, but that do not reflect the true distribution of the data. In this way, our algorithms can overfit the unlabeled data. With significantly more unlabeled data, this problem would be alleviated, but in many domains, only a finite amount of unlabeled data may exist.

When presented with this scenario, unrepresentative generative models can exacerbate the problems of overfitting. For example, consider the three-way classification task in Figure 3.6. The frequency of generic cooking words like *skillet* and *grill* are likely to be very similar in the Italian food and Mexican food classes. However, given our basic generative model, these classes are assumed to be independent; thus these word frequencies must be estimated for each class using its available data. Each estimate will be different, matching the peculiarities of the documents in its class. If our model instead noted the relationship between them, the training data for these classes could be pooled together to form more accurate estimates for these generic cooking word frequencies.

By using a more representative model that encodes hierarchical class relationships, we help ensure that models with high-likelihood on the unlabeled data are also high-likelihood models on the true data distribution. When this relationship holds, this helps parameter estimation find high-accuracy classifiers.

In Section 3.3.1 we show how to learn high-likelihood parameters of a hierarchical model with EM. In Section 3.3.2 we present the Cora dataset, a collection of computer science research papers that comes with a class hierarchy that describes the sub-fields

- **Inputs:** A collection of unlabeled training documents, a class hierarchy, and a few keywords for each class.
- Generate initial labels for as many of the unlabeled documents as possible by term-matching with the keywords in a rule-list fashion.
- Split the data into two disjoint subsets, $\mathcal{D}^w$ and $\mathcal{D}^s$.
- Initialize the shrinkage weights $\mathrm{P}(a_n|c_j;\theta)$, to be uniform along the path from each leaf node to the root of the hierarchy.
- Estimate word probability and class prior parameters using the initially labeled data (Equations 3.11 and 3.9).
- Loop until parameter convergence:
    - **(E-step)** Use the current classifier to estimate the class labels of each document (Equation 3.6). Accumulate the ancestor word generation counts using only documents in $\mathcal{D}^s$ (Equation 3.7) .
    - **(M-step)** Re-estimate the shrinkage weights by normalizing the ancestor word counts (Equation 3.8). Re-estimate the class priors and word probability parameters using only documents in $\mathcal{D}^w$ (Equations 3.11 and 3.9).
- **Output:** A text classifier that takes an unlabeled document and predicts a class label using Equation 3.6.

Table 3.7: The modified outline for likelihood maximization from unlabeled data using a hierarchical model and EM.

of CS. Section 3.3.3 demonstrates that by using a more representative model, we learn parameters from unlabeled data that increase classification accuracy, where with the basic model, accuracy decreases. Section 3.3.4 discusses the implications of these results.

3.3.1 Estimating Parameters of a Hierarchical Model

We leverage the hierarchy through a technique known as *shrinkage* (Carlin & Louis, 1996). Hierarchical shrinkage calculates word probability estimates for each leaf class by calculating a weighted average of the estimates along the path from the leaf to the root. This technique balances a trade-off between specificity and reliability. Estimates in the leaf are most specific but unreliable; further up the hierarchy estimates are more reliable but unspecific. These mixture weights can be set by EM at the same time that the word probability parameters are being set by EM, both using labeled and unlabeled data. This joint EM process is described in Figure 3.7.

One can think of hierarchical shrinkage as a generative model that uses the hierarchy. Every leaf in the hierarchy is a class. First, pick a class with a biased die roll based on class priors. Then pick a document length (independently of class). When generating words for the document, some will be very generic words and others will be class-specific. Thus, choose one hierarchy ancestor of the class node (along the path from the root to the leaf, including possibly itself) according to the shrinkage weights. Choose *one* word from the multinomial of the selected ancestor. Repeat this process of choosing an ancestor and picking a word for each position in the document. In this way, the document created will be a mixture of specific and general words that are drawn using the hierarchical relationships provided.

Let us introduce some notation to describe hierarchies. As originally, let c_j be a class; there is one class for each leaf node of the hierarchy. Let any node in the hierarchy be denoted by a_n. Note that we make a distinction between a class and a node. A leaf in the hierarchy has both a node and a class. The *ancestors* of a class, $\text{ancs}(c_j)$, include all nodes on the path from a leaf to the root, inclusive of both. Conversely, the set $\text{leaf}(a_n)$ contains all class leaf nodes below (and possibly including) node a_n in the hierarchy. The hierarchical shrinkage weights we denote with $\text{P}(a_n|c_j;\theta)$. Note the shrinkage weights are dependent on the class, implying that each class has a set of weights over all nodes in the tree. However, the weight of any node that is not an ancestor of the class is always zero.

If we ran EM to set both the word probabilities and the shrinkage weights using the same set of data, the shrinkage weights would concentrate in the leaves. This would happen because the most-specific model would best fit the training data. This would result in exactly the overfitting we are trying to prevent. To avoid this problem, we split our training data into two disjoint sets. We use the documents in subset $\mathcal{D}^w$ to estimate the word probability and class prior parameters. For the shrinkage weights, we use subset $\mathcal{D}^s$. We concurrently perform EM to set the shrinkage weights with EM to set the multinomials.

In the E-step, we must estimate the class label of each unlabeled document. Calculating class probabilities is the same as in previous sections:

$$\mathrm{P}(c_j|d_i;\theta) = \frac{\mathrm{P}(c_j|\theta)\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{i,k}}|c_j;\theta)}{\sum_{r=1}^{|\mathcal{C}|}\mathrm{P}(c_r|\theta)\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{i,k}}|c_j;\theta)}. \tag{3.6}$$

In the E-step for the shrinkage weights, we accumulate counts of how many words in each class would have been generated by each ancestor node. We accumulate these counts, $N(a_n, c_j)$ over just the documents in subset $\mathcal{D}^s$:

$$N(a_n, c_j) = \frac{\sum_{i=1}^{|\mathcal{D}^s|}\sum_{k=1}^{|d_i|}\mathrm{P}(c_j|d_i;\theta)\mathrm{P}(a_n|c_j;\theta)\mathrm{P}(w_{d_{i,k}}|a_n;\theta)}{\sum_{i=1}^{|\mathcal{D}^s|}\sum_{k=1}^{|d_i|}\sum_{a_m\in\mathrm{ancs}(c_j)}\mathrm{P}(c_j|d_i;\theta)\mathrm{P}(a_m|c_j;\theta)\mathrm{P}(w_{d_{i,k}}|a_m;\theta)}. \tag{3.7}$$

For the M-step, we take these estimates of the class labels of each document and the counts of the word sources and calculate new parameter estimates that will be of higher likelihood than before. Up until now, we have always used maximum a posteriori parameter estimation. The motivation for this was to prevent word probabilities of zero for infrequently occurring words. Here, we will accomplish this in a different way. We augment the given hierarchy by placing a new root node on top of the old one. We permanently fix the word probabilities of this root node to be uniform across the vocabulary. This uniform multinomial ensures that all word probabilities, when mixed over the path from a class to the root, are non-zero. Thus, with this extra hierarchical twist, we can use maximum likelihood estimation instead of maximum a posteriori estimation. One benefit of using this approach is that each class determines for itself the best amount of smoothing. Before, with Laplace smoothing, we fixed the Dirichlet prior whereas now the data determines how much smoothing is needed. This effect is true for the shrinkage weights at all levels of the hierarchy. For classes with a lot of data, or word distributions very different from its ancestors, we would expect the shrinkage weights to favor the leaf node. But for classes with very sparse training data we would expect significantly higher dependence on the shared ancestors, including the uniform root node.

Calculating the new shrinkage weights is quite easy. We simply take the accumulated counts and normalize them into probabilities:

$$\mathrm{P}(a_n|c_j;\theta) = \frac{N(a_n, c_j)}{\sum_{a_m \in \mathrm{ancs}(c_j)} N(a_m|c_j;\theta)}. \tag{3.8}$$

Estimating the class probabilities is done just by summing up the fractional memberships over all documents, exactly as we did with the original generative model:

$$\mathrm{P}(c_j|\theta) = \frac{1 + \sum_{i=1}^{|\mathcal{D}^w|} P(c_j|d_i;\theta)}{|\mathcal{C}| + |\mathcal{D}|}. \tag{3.9}$$

We calculate the new word probabilities in two steps. First, using only documents in $\mathcal{D}^w$, we calculate the ancestor node word probabilities by pooling the counts from its leaf classes, and calculating maximum likelihood estimates:

$$\mathrm{P}(w_t|a_n;\theta) = \frac{\sum_{i=1}^{|\mathcal{D}^w|} \sum_{c_j \in \mathrm{leaf}(a_n)} N(w_t, d_i)\mathrm{P}(c_j|d_i;\theta)}{\sum_{s=1}^{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{D}^{\sqsupseteq}|} \sum_{c_j \in \mathrm{leaf}(a_n)} N(w_s, d_i)\mathrm{P}(c_j|d_i;\theta)}. \tag{3.10}$$

Then, to calculate our new word probability parameters for each class, we take a weighted average over the ancestors using the shrinkage weights:

$$\mathrm{P}(w_t|c_j;\theta) = \sum_{a_n \in \mathrm{ancs}(c_j)} \mathrm{P}(a_n|c_j;\theta)\mathrm{P}(w_t|c_j;\theta). \tag{3.11}$$

That completes the explanation of the EM iterations. In the dataset used below, no labeled data are available during parameter estimation with EM. This makes the EM initialization process quite different then before. Instead of labeled data, for the domain in question, we are provided with a small number of key words and phrases for each class. We use these keywords to assign initial labels to some of the unlabeled documents by term-matching in a rule-list fashion: for each document, we step through the keywords and place the document in the category of the first keyword that matches. If a document matches no keywords, it is not used during initialization. We treat these initial labels as our class membership estimates. We initialize the shrinkage weights to be uniform along the path from each leaf to the root. Using the initial labels and the shrinkage weights we calculate the word probability parameters. This gives us an initialization for EM. After the first priming M-step, these initial labels are discarded and are replaced by the E-step estimates in each iteration thereafter. In this way, we use limited domain knowledge given by the keywords to provide a reasonable initialization for EM.

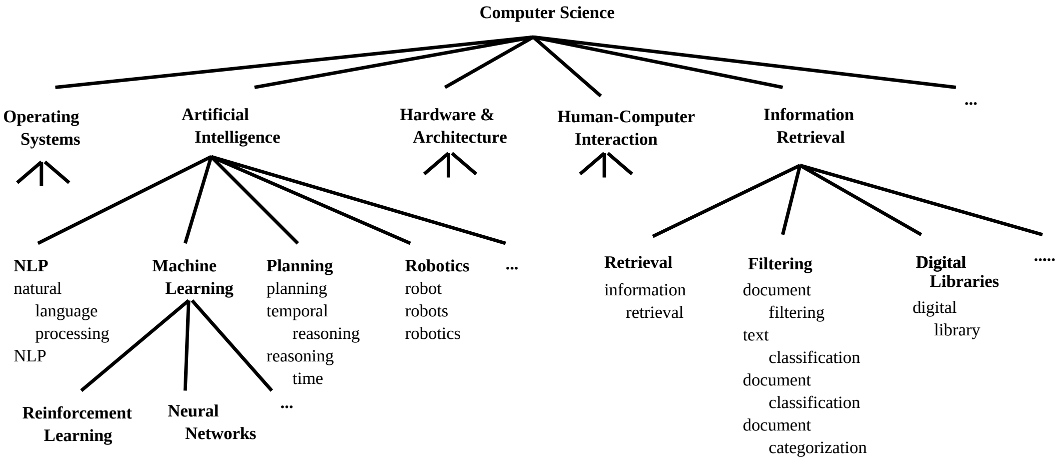

Figure 3.7: A subset of Cora's computer science hierarchy with the complete keyword list for each of several categories. These keywords are used to find unlabeled documents to initialize EM. The complete hierarchy and keywords are detailed in Appendix A.

3.3.2 Dataset and Protocol

As a test domain for our hierarchical generative model we use the Cora dataset, a collection of computer science research papers classified into 70 sub-fields of CS. This dataset is used to build text classifiers to automatically place research papers spidered from the Web in postscript format into their appropriate sub-field. This taxonomy, along with a search engine over the papers and an automatically-constructed citation graph, is a publicly-available resource for researches and practitioners (McCallum et al., 2000). To support the taxonomy, there is a hierarchy of computer science topics, part of which is shown in Figure 3.7. The hierarchy was created by examining conference proceedings and computer science sites on the Web. A small test set was created by expert hand-labeling of a random sample of 625 research papers from the 30,682 papers in the Cora computer science archive at the time of these experiments. Of these, 225 (about one-third) did not fit into any category and were discarded—resulting in 400 labeled documents. Some of the discarded papers were outside the area of computer science (e.g., astrophysics papers), but most of these were papers that with a more complete hierarchy would be considered computer science papers. The class frequencies of the data are skewed, but not drastically; on the test set, the most populous class accounted for only 7% of the documents.

Each research paper is represented as the words of the title, author, institution, references, and abstract. These segments are automatically extracted using a trained hidden Markov Model. The extraction is performed independently of the classification task and is described in detail by Seymore et al. (1999). Words occurring in fewer than five documents and words on a standard stoplist are discarded. No stemming is used.

Since only 400 documents have labels, we use all of these documents as a test set to measure classification accuracy. This does not leave any documents available for labeled training data. As a replacement for labeled data, we use several human-provided key words and phrases for each class to set the starting point of EM. Keywords are much quicker to generate than even a small number of labeled documents. We use keywords to generate initial labels for some unlabeled documents; these initial labels are used to set the EM starting point. Figure 3.7 shows examples of the number and type of keywords selected for our experimental domain. The complete hierarchy and the keywords are detailed in Appendix A.

Since we provide only a few keywords for each class, classification by keyword matching is both inaccurate and incomplete. Keywords tend to provide high-precision and low-recall; this brittleness leaves many documents unlabeled. Some documents match keywords from the wrong class. In our experimental domain, for example, 59% of the unlabeled documents do not contain any keywords. Among documents containing keywords, the precision of the keyword matching on the test set is 75%.

The complete algorithm used for the hierarchical experiments is given in Table 3.7. In our experiments we use 3000 randomly-selected documents to estimate the shrinkage weights and the remaining to estimate the word probabilities. Many fewer documents are needed to accurately estimate the relatively small set of shrinkage parameters.

3.3.3 Experimental Results

In this section, we provide empirical evidence that using EM with unlabeled data and a hierarchy produces a high-accuracy text classifier. Table 3.8 shows classification results with different classification techniques used. Two interesting baselines provide a sense of the difficulty of the classification task. With 399 labeled documents (using our test set in a leave-one-out-fashion), naive Bayes reaches 47% accuracy. The test set was also relabeled by a second human expert; the agreement with the original labeling was only 72%, showing that the classification task in question is a challenging one.

Method	# Labeled	# Initially Labeled	# Unlabeled	Accuracy
Naive Bayes	399	—	—	47%
Human Agreement	—	—	—	72%
Keyword Matching	—	—	—	46%
Naive Bayes	—	12,657	—	63%
EM: shrinkage only	—	12,657	—	63%
EM: w/o hierarchy	—	12,657	17,625	58%
EM: with hierarchy	—	12,657	17,625	66%

Table 3.8: Classification results with different techniques: keyword matching, naive Bayes, and EM with and without a hierarchical generative model. The classification accuracy, and the number of labeled, keyword-matched initially labeled, and other unlabeled documents used by each variant are shown. Best algorithmic performance is achieved using all the unlabeled data with a hierarchical generative model.

We begin by using keywords to initially label some documents for the EM starting point. The keywords themselves, when applied to the test set, give 46% accuracy.[2] When applied to the unlabeled data, the keywords generate initial labels for 12,657 out of 30,282 documents. If we use these initially labeled documents to build a naive Bayes classifier treating the labels as correct, we get 63% accuracy. If we fix these labels, and run only EM for shrinkage, ignoring the remaining documents, accuracy does not increase further. If we use the original generative model without the hierarchy, EM finds parameters with higher likelihood on the unlabeled data. However, classification accuracy decreases to 58%. Thus, likelihood maximization for the naive Bayes model does not do well for this complex classification task with sparse data.

When we switch to our hierarchical model, we see that now likelihood maximization increases classification accuracy. Hierarchical EM with all the unlabeled data improves accuracy to 66%. This shows that by using a more representative generative model, we are able to overcome the negative effects of overfitting.

3.3.4 Discussion

The most interesting experimental result is the difference in performance between EM with and without the hierarchy. Without the hierarchy likelihood maximization

[2] The 43% of documents in the test set containing no keywords are not assigned a class by the rule-list classifier, and are assigned the most populous class by default.

with EM lowers classification accuracy. However with the more representative model, increasing likelihood with EM also increases classification accuracy. With a less representative model and sparse data, models with higher likelihood on the unlabeled data do not correspond to higher accuracy classifiers, due to overfitting.

It is interesting to notice that hierarchical shrinkage on its own does not improve the performance of the data. With only labeled data classification accuracy does not increase with a better model. This provides further evidence that learning from labeled and unlabeled is more sensitive to the match between the data and the model than is regular supervised learning without unlabeled data.

Hierarchical models for text have been used in the literature for both unsupervised clustering and for purely supervised clustering from labeled data. McCallum et al. (1998) use hierarchical shrinkage to form more reliable parameter estimates from sparse labeled training data for text classification. They show improved classification accuracy for datasets with a hierarchy. Hofmann and Puzicha (1998) used a very related model to create a hierarchy in an unsupervised fashion from a collection of scientific abstracts. The work in this chapter is the first to apply these models to learning from a combination of labeled and unlabeled data.

In this chapter we changed two of the three generative assumptions. Other studies have examined relaxing the word independence assumption for supervised learning from labeled data. In the context of a multi-variate Bernoulli generative model, Sahami (1996) allows for limited word dependencies within each class. Specifically he allows each word to depend on exactly one other, in essence creating a dependency tree. He finds that classification performance with this model is higher than a strict word independence model.

The multinomial distributions we use are equivalent to unigram language models. Mladenic and Grobelnik (1999) explore using a more sophisticated bigram language model for classification. In this model, a word occurrence probability depends on the word previous to it in the document. Their findings show that this bigram modeling gives better classification performance.

Other work (Li & Yamanishi, 1997) relaxes the one-to-one correspondence differently then here. They allow instead for a one-to-many correspondence between clusters and classes, instead of a many-to-one relationship. They use the same number of mixture components as there are classes, but they introduce a probabilistic relationship between components and classes.

The models of this chapter can be extended in several natural ways. The use of multiple mixture components per class and hierarchical generative models can be combined. This approach would be suggested when a hierarchy is provided, and each class in the hierarchy is complex and multi-faceted. The hierarchical model can also be naturally extended to model directed acyclic graphs, instead of restricting class relationships to be expressed with tree structures.

In summary, this chapter has demonstrated that when using the generative model approach, it is necessary to use a representative model. If model probability and accuracy are not well-correlated, then the use of unlabeled data will hurt classification, instead of help. Often, if needed, the generative model can be improved. We have shown two different ways of making models more representative, and have demonstrated that with them, unlabeled data can be successfully used in learning text classifiers.

Chapter 4

Finding More Probable Models

We have demonstrated that by assuming a generative model we can incorporate unlabeled data into supervised learning to improve the accuracy of text classification. This chapter provides two ways to improve performance with unlabeled data even further when the generative model is representative and labeled data are sparse. Under these conditions, the number of *labeled* examples is a bottleneck for more significant improvements. This bottleneck is caused when the sparse labeled data provide a poor EM initialization that results in a low probability local maximum. Traditionally this is mitigated by running EM many times from different random starting points. For text domains, the best classifier from many random initializations is not better than the one initialized deterministically from the labeled data. The parameter space is too large to randomly explore for good initializations. One alternative technique is to use limited interaction with a human labeler; then specific documents can be selected for labeling by the learning algorithm. This allows the learner to influence the EM initialization. We use a Query-by-Committee approach to active learning that requests labels for documents that are prototypical but have high classification variance. Experimentally, data labeled in this fashion provide a higher accuracy initialization, and after EM, a more accurate classifier. Another way to improve on the weakness of EM is to consider other maximization techniques. Deterministic annealing is a technique that avoids local maxima by maximizing first on a very smooth probability surface and gradually making it more bumpy, tracking the maximum as the surface gets more complex. Experimentally, deterministic annealing finds more probable models (and thus higher-accuracy classifiers) when labeled training data are sparse.

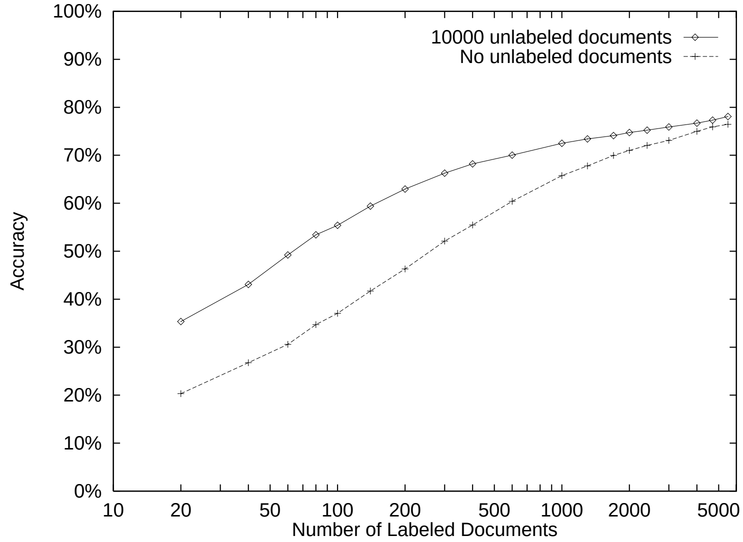

Figure 4.1: Classification accuracy on the 20 Newsgroups data set, both with and without 10,000 unlabeled documents. Using unlabeled data helps, but when labeled data are sparse, there is still significant room for improvement.

4.1 Introduction

In the previous chapter, we saw several datasets that did not make good use of unlabeled data with the basic algorithm of Chapter 2. When the generative model assumptions were strongly violated by the true data distribution, parameters with high probability did not correspond to classifiers with high accuracy. By modifying the assumed generative model to be more representative of the data, classification accuracy and model probability came into correspondence. Integrating the evidence of the unlabeled data with this improved model allowed us to find classifiers with higher accuracy.

In this chapter we explore how to further improve learning performance when classification accuracy and model probability are *already* in good correspondence. For example, in Section 2.6.2 we presented experimental evidence for the 20 Newsgroups dataset showing that (1) classification accuracy and model probability are in strong correspondence, and (2) integrating unlabeled documents through posterior model maximization improves text classification accuracy. When labeled data were sparse the improvements were the largest, but performance was still substantially below that achieved with plentiful labeled data. These findings, originally shown in Figure 2.2, are shown again here in Figure 4.1.

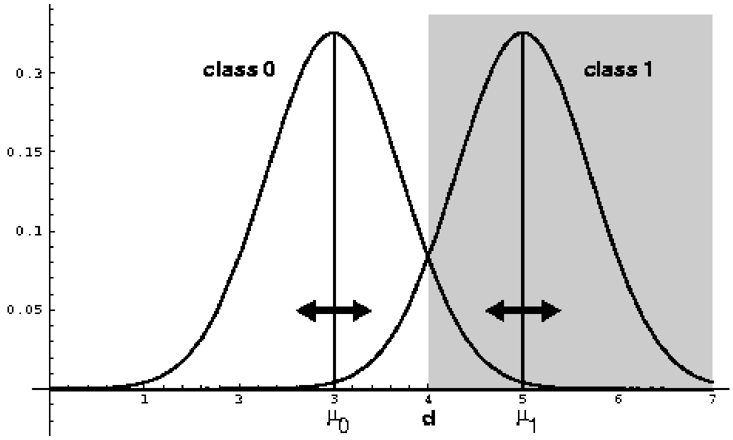

Figure 4.2: With an infinite amount of unlabeled data, the true parameters of the generative model can be recovered. Then, with just a few labeled examples, each class can be assigned to its cluster, resulting in the Bayes-optimal classifier.

Sometimes having only sparse labeled data does not matter. For example, consider the case of estimating two univariate normal distributions for classification. With an infinite amount of unlabeled data we will recover the global maximum a posteriori parameters for each mixture component, as indicated in Figure 4.2. Then, using just a few labeled examples, we can correctly match up classes to components and have the Bayes-optimal classifier (Castelli & Cover, 1995). Things are much different in reality where we have a large but not infinite amount of unlabeled data and high-dimensional feature spaces. Here, local maxima abound in the probability surface, and we cannot easily find the global maximum. However, this is *not* reason to be resigned to weak performance with sparse labeled data; there could be more juice to be had from the unlabeled data. By modifying our algorithms, we may be able to make more effective use of the unlabeled data on hand.

This chapter explores ways to improve performance when the generative model is representative but labeled data are limited. In these cases performance suffers from getting stuck in local maxima during the EM search in model probability space. Section 4.2 shows this is primarily caused by the poor EM initializations given by the labeled data. Section 4.3 shows that the standard approach of multiple EM runs with random initialization is unlikely to be helpful for text classification. In Section 4.4 an active learning algorithm, in conjunction with a labeler, provides improved initializations that lead to classification accuracy increases. Section 4.5 demonstrates that

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	79	1	1	.	.	1	2	3	1	.	.	.	.	.	4	88	4	4	3	9
1	1	43	110	1	.	7	9	3	3	.	.	4	.	.	11	4	.	1	3	.
2	1	2	176	.	1	1	5	2	.	2	.	2	.	.	.	4	1	.	3	.
3	.	7	100	18	15	.	52	4	1	.	1	.	.	.	1	.	.	.	1	.
4	.	3	79	4	33	4	58	7	1	.	.	.	.	.	1	3	.	1	6	.
5	.	13	142	.	1	33	.	1	1	.	.	1	.	.	5	1	.	.	2	.
6	.	3	14	1	4	1	136	12	4	.	2	1	2	1	7	2	3	1	6	.
7	1	.	2	.	.	1	6	168	7	.	.	.	3	.	2	1	1	.	7	1
8	.	.	1	.	1	.	2	158	33	.	.	.	1	.	.	.	.	.	3	1
9	.	1	1	.	.	1	2	2	.	127	55	.	.	.	1	1	.	2	7	.
10	.	2	.	.	.	.	.	.	.	1	186	1	.	.	1	3	.	.	6	.
11	.	1	15	.	.	1	.	3	1	.	1	160	.	1	5	1	7	.	2	1
12	2	10	32	3	15	1	17	58	4	3	1	6	21	.	22	1	.	.	4	.
13	4	11	2	1	.	4	1	16	9	.	2	.	2	35	15	27	6	1	64	.
14	2	2	1	.	1	.	2	3	1	1	.	3	1	.	153	8	2	.	20	.
15	.	.	2	1	.	2	.	.	.	.	1	.	.	.	1	186	.	.	1	3
16	1	5	.	.	1	.	1	2	1	.	.	2	.	2	.	30	139	3	13	.
17	.	.	.	.	.	.	.	1	1	2	.	.	.	.	1	10	.	181	3	1
18	6	1	1	.	.	.	.	4	1	.	1	2	.	.	7	63	59	11	41	3
19	42	2	1	.	.	.	.	2	1	.	.	2	.	.	3	112	12	3	7	13

Table 4.1: One randomly selected confusion matrix for the test set with 60 labeled documents (3 per class) for 20 Newsgroups. True classes are in rows, and predicted clusters are in columns. The preponderance of documents lying along the diagonal indicates that the class-to-cluster correspondence is not a significant factor in classification accuracy.

deterministic annealing, a maximization algorithm similar to EM, achieves higher model probability and classification accuracy by local maxima avoidance.

4.2 The Influence of Labeled Examples

Understanding the role labeled data play in our algorithm will help us see why performance could be better when labeled data are sparse. When learning with labeled and unlabeled data, the labeled data influence the result of our algorithm in three ways: they (1) correlate each cluster with a class, (2) influence parameter estimation during the EM iterations, and (3) initialize the starting point of EM. In this section we examine the relative strengths of these influences, and conclude that the initialization is the most important role played by the labeled data.

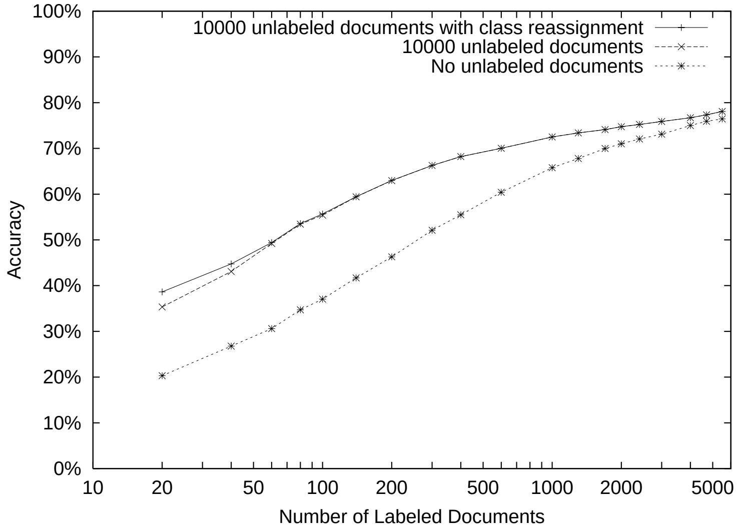

Figure 4.3: Classification accuracy on the 20 Newsgroups data set with labeled and unlabeled documents using class reassignment. The benefits of reassignment are minimal, indicating that faulty class correlation is not a significant factor for classification.

Even when EM finds a high-probability model, classification can be very poor if we simply swap the class identities around among the clusters. For example a classifier that perfectly distinguishes between sports and arts will suddenly get 0% accuracy if we swap their definitions. In our experiments clusters are correlated with classes by assigning each cluster to the class of the documents it was initialized with. Thus, we make the implicit assumptions that (1) the initialization places each cluster in the right neighborhood for its class, and (2) each cluster successfully tracks the same class through the EM iterations. In contrast to this approach, the correlation process could also be done after iterations are complete, by seeing into which cluster each class's labeled documents fall. When the clustering is perfect this method requires only a very small amount of labeled data to correctly match up each cluster with its class (Castelli & Cover, 1995).

Is this correspondence happening poorly for us when labeled data are sparse? Table 4.1 shows a typical confusion matrix for 20 Newsgroups with only three labeled documents per class. In all but one case, each cluster is assigned to the class that has the plurality of its documents. Anecdotally, the diagonal of our confusion matrices has always been dominant in this way. We can measure our loss in accuracy due to incorrect class correspondence by greedily reassigning classes to maximize accuracy

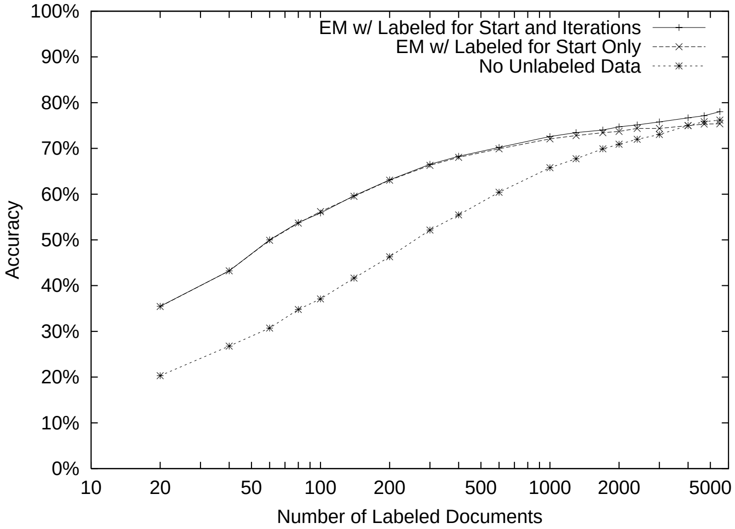

Figure 4.4: A comparison of EM performances, with differing use of the labeled data. The top line shows normal EM with labeled and unlabeled data. The second line, almost overlapping the first, shows performance of EM when the labeled data is used only to set the initial parameters for EM. Their strong similarity indicates that the power of the labeled data comes not from its use during the iterations, but in setting the initial parameters.

based on the performance of the test data.[1] Figure 4.3 shows these results; they indicate only a minimal loss from poor reassignment. For example, with only two labeled documents per class (40 total), class correspondence problems lower accuracy from 45% to 43%. From this we deduce that classification accuracy with a small amount of labeled data does not suffer significantly due to class correspondence problems.[2]

Let us now consider the effect of the labeled data during the iterations of EM. One way to think about our application of EM is that it performs semi-supervised clustering. The labeled data provide supervision during the iterations of EM by remaining fixed to their correct class. However, with several orders of magnitude more unlabeled documents than labeled ones, the great majority of the data comes from the unlabeled set. Thus during the EM iterations, we would expect the class mixture components to be mostly positioned to maximize the likelihood of the unlabeled documents.

We can explicitly measure the effect labeled data has during these iterations with

[1] To optimally reassign classes is not feasible for a 20 class problem, as it would involve considering all permutations. For smaller classification problems, this would not be a factor.

[2] For a problem with a huge number of classes, this will likely become a more significant factor, as the number of possible errors grows.

a simple experiment. We test this effect by performing our regular algorithm, but withholding the labeled data during the EM iterations. This compares the effects of using the labeled data for just for the initialization to using it for both the initialization and the iterations. The results of this experiment for 20 Newsgroups are shown in Figure 4.4. The accuracy of classifiers built without labeled data in the iterations is essentially the same as our typical algorithm especially when labeled data are sparse. For example with two labeled documents per class accuracy is 43% for both cases. The curves only start to (marginally) diverge when there is a large amount of labeled data. This indicates that the influence of the labeled data during the EM iterations is quite minimal.

These two simple experiments have eliminated two of the three possible effects of the labeled data—class-to-component correspondence and influence during EM iterations. From this we deduce that limited labeled data hinder the use of unlabeled data in our approach primarily by giving poor EM initializations.

Why would the initialization have so much effect? One possible explanation is that with a more probable initialization, many poor local maxima will be completely bypassed. That is, since model probability can only increase with EM, any initialization successfully avoids all local maxima with a lower probability. A second explanation could be that there is regularity in the probabilities of the local maxima. An initialization may direct EM towards regions in parameter space that tend to have higher maxima. Some optimization algorithms in other domains focus explicitly on this effect and model the regularities of the local maxima to select good initializations (Boyan, 1998).

There are three different ways to address the problem that limited labeled data give poor EM initializations. First, we could choose the initializations in a different way, without a strong dependence on the labeled data. Second we could continue using labeled data to initialize EM, but ensure that the labeled data are of high-quality. The third idea is to dispense with EM and use a different algorithm for maximizing the model posterior that is not so sensitive to the labeled data. These three directions are addressed by the following sections in turn. The next section explores using random initialization instead of using labeled data deterministically. Section 4.4 uses active learning to select high-quality labeled data. Section 4.5 uses deterministic annealing instead of EM to avoid poor local maxima.

4.3 Using Many Random Starting Points

In the previous section it was demonstrated that when learning text classifiers the labeled data have their primary influence when they are used to form the initial parameter estimates for EM. When labeled data are plentiful, the initial parameters have high probability and accuracy; EM can then incorporate the evidence of the unlabeled data and output a high-accuracy classifier. On the other hand, when labeled data are sparse the initial parameter estimates will have a much lower probability. As a result EM gets caught in a local maximum that gives improved accuracy, but significantly below what could be obtained with a better initialization.

In our algorithm, we have always initialized the parameter estimates to the MAP estimates derived from the labeled data alone. This initialization places the classifier into an approximate neighborhood of parameter space that corresponds to document and class distributions. In many other applications that use the EM framework there is no analog of the labeled data from which to set the starting point. In these cases, EM is typically initialized with *random* parameter estimates.

A single random parameter initialization frequently results in a poor local maximum because the parameter space has so many local maxima in which only a small minority are of high probability. To overcome this, it is standard practice to run EM many times from many different random initializations. Given the set of all EM results, the one with the highest model probability is then selected as the best parameterization. Multiple random runs allow an exploration of the local maxima and provide a robustness against any one poor parameter initialization.

We apply this same approach to our text classification task. When using unlabeled data, setting the starting point deterministically with only a few labeled examples does help classification. But, there is still room for improvement. The hope is that by randomly selecting many starting points we will find better a better initialization than the deterministic one, and produce a higher-accuracy classifier. However, it may be hard in practice to find a better initialization because the labeled data provide significant information on their own.

4.3.1 Choosing Random Starting Points

There are many ways of using randomness for parameter initializations. For example, one possibility would be to set all word probability parameters to some small perturbation of the uniform distribution. If nothing were known about the domain, this might be a reasonable approach. For our situation, though, this would be a poor implementation choice, because this setting does not represent any basic knowledge of text. For example, if one mixture component had significantly higher-than-average probabilities for a few very common words such as *the* or *and*, then it is likely that in the first round of EM *all* unlabeled documents would be classified as belonging to that cluster. Running EM from this initialization would give a terrible classification model. Some of our preliminary experiments with random initializations showed that this consistently got very low probability models.

In text domains we already have some good guesses about some of the word probabilities by looking at the limited labeled data and the vast unlabeled data. Many words will not have large differences in their probabilities across classes; a class-unconditional estimate would be a fair approximation. Instead of using a uniform distribution as the baseline we can use a mixture of the word frequencies in the labeled and unlabeled data as our baselines. Specifically, for our experiments we use a priming E-step to assign uniform classification posteriors to each unlabeled document. Then, we set the baseline of each initial component to the MAP parameter estimate from the limited labeled documents and the spread-out unlabeled ones. This baseline accurately represents both the knowledge from the labeled data, as well as significant influence from the unlabeled.

To introduce randomness into this baseline, we need to perturb these estimates appropriately. How do we do this? We can again use our generative model approach. Given the labeled and uniformly-spread unlabeled data, our baseline for each class is the MAP estimate, $\arg\max_{\theta_j} \mathrm{P}(\theta_j|\mathcal{D})$. In other words, the training data specifies a probability distribution over the parameters of the mixture components; until now we have used the most probable parameterization from that distribution. We can introduce randomness by selecting our parameters according to the probability distribution given the labeled and unlabeled data, instead of just choosing the most probable one.

Using the notation introduced in Chapter 2, the data define a Dirichlet distribution over the multinomial parameters of each class:

$$\mathrm{P}(\theta_j|\mathcal{D}) \propto \prod_{t=1}^{|V|} \mathrm{P}(w_t|c_j;\theta)^{\alpha_t - 1} \tag{4.1}$$

where α_t is the smoothed number of word occurrences seen in the training data for that class:

$$\alpha_t = \sum_{d_i \in \mathcal{D}} \mathrm{P}(c_j|d_i) N(d_i, w_t) + 2. \tag{4.2}$$

The Dirichlet distribution is the commonly-used conjugate prior distribution for multinomials. A good intuitive introduction to Dirichlet distributions is given by Stolcke and Omohundro (1994).

Sampling from a Dirichlet distribution is relatively straightforward. This is performed by drawing weights, v_{tj}, for each word w_t and class c_j from the Gamma distribution: $v_{tj} = \mathrm{Gamma}(\alpha_t)$. Then we set the parameters $\theta_{w_t|c_j}$ to the normalized weights by $\theta_{w_t|c_j} = v_{tj} / \sum_s v_{sj}$. Sampling from the Gamma distribution is also not hard; one method is detailed by Press et al. (1993).

Thus, we can initialize our parameters randomly by using the generative model assumptions. This enables us to choose parameters that fall within the reasonable range of text space but also provide a generous amount of variation. These random initializations are used by EM to find different local maxima of which at least one hopefully will give a high probability (and thus high accuracy) model.

4.3.2 Dataset and Protocol

For the remainder of this chapter, we use a subset of the 20 Newsgroups dataset. This subset, News5, is the five confusable comp.* classes (comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, and comp.windows.x). The dataset contains 4982 documents, nearly evenly divided among the five classes. The data are pre-processed the same as 20 Newsgroups, as described in Section 2.6.1. Briefly, we use a stoplist, do not stem, and ignore all the UseNet headers, even Subject and From.

In order to provide comparable model probability numbers, we fix a single vocabulary for all our experiments. We use the top 4000 words by mutual information

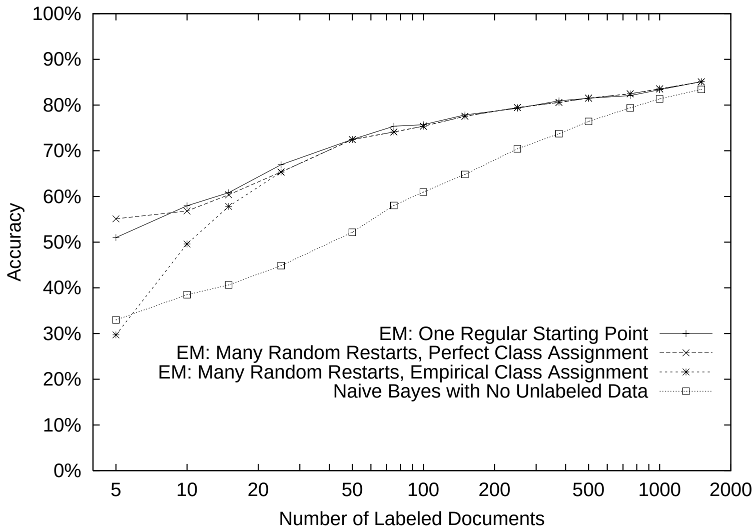

Figure 4.5: A comparison of different techniques for choosing a starting point. The best performance is achieved with a single deterministic initialization from the labeled data. When the most probable model from many random initializations is used, accuracy is worse than before when labeled training data is sparse. However, if classes and clusters are matched up perfectly, results are about the same as the deterministic initialization.

measured over the entire labeled dataset. Chapter 6 discusses the issue of feature selection for learning from labeled and unlabeled data.

In our experiments, 600 random documents per class (3000 total) are treated as unlabeled. A fixed number of labeled examples per class are also randomly selected. The remaining documents are used as a test set. For each experiment with a fixed number of labeled examples, we perform ten random test/train/unlabeled splits. These splits are paired across all conditions. That is, the exact same splittings are used to compare each algorithm.

When performing multiple random restarts, we run EM 100 times per split, and select the run with the highest model probability. Each random starting point is chosen as described in Section 4.3.1. Out of these 100, the model with the highest probability is selected as the chosen model for which to measure classification accuracy on the test data.

4.3.3 Experimental Results

Figure 4.5 shows the accuracies achieved by using multiple random restarts. With at least a moderate amount of labeled data random initialization performs at essentially the same accuracy as deterministic initialization. With just a few training examples per class, however, accuracy of random initialization is significantly worse than deterministic initialization. For example, with two labeled documents per class (10 total), random initialization finds a model with 50% accuracy; the single deterministic initialization get a model with 58% accuracy. One problem with choosing the starting points randomly is that the randomness can overcome the signal in the labeled data (remember that labeled data are used in choosing the random initialization). This problem is worst when there are only a small amount of labeled data; then, the baselines for each the mixture component are quite close to each other. Random variation can easily throw off the class-to-component correspondence.

Measuring the strength of this class-correspondence effect is easy. We make use of the test data to reassign classes to components to maximize classification accuracy on the test set. This way we can identify cases where, for example, the data cluster for the comp.graphics.misc class is actually represented by the mixture component initialized to be comp.windows.x. The results of this analysis are also shown in Figure 4.5, demonstrating an upper-bound on the performance of our random initialization. The accuracy of random initializations with optimal class assignment is essentially the same as deterministic initialization across all labeled set sizes. For example with two labeled documents per class we now improve from 50% to 57%, compared to 58% for a single initialization. This indicates that random initializations will not dramatically improve performance when labeled data are sparse, even if we could perfectly correlate clusters and classes.

One might think about what these results suggest about our belief that model probability and accuracy are correlated. The multiple random initialization approach could be finding more probable models then the deterministic approach. After all, it has 100 chances to find different local maxima. If indeed this were so, then we would wonder whether model probability and accuracy were so strongly correlated after all. But by analyzing the results, we find that it is not the case that random initializations find more probable models.

Figure 4.6 shows a scatterplot indicating the relationship between model probability and accuracy for both random and deterministic initializations. We show two

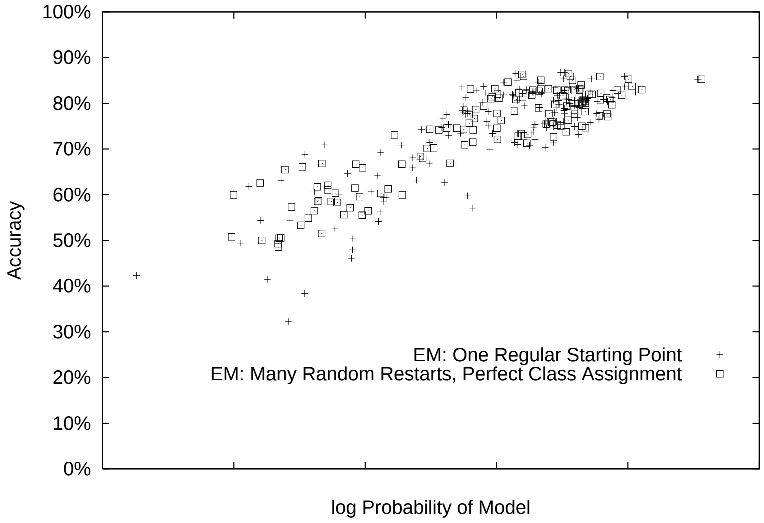

Figure 4.6: A scatterplot showing the relationship between model probability and accuracy for both random initialization and deterministic initialization from labeled data. Both cases are distributed similarly, reinforcing the evidence that model probability and accuracy are strongly correlated.

points for each test/train/unlabeled set used in Figure 4.5—one indicating the result of the deterministic starting point and the other showing the best of many random initializations. Two findings are of interest. First, we again see a strong correlation between accuracy and model probability, as seen with the 20 Newsgroups dataset. This is not too surprising because News5 is a subset of 20 Newsgroups, and shares many properties with it. More importantly, note that the random initializations have approximately the same distribution as deterministic initializations. Both have the same correlations between accuracy and model probability. This indicates that the relationship between accuracy and model probability holds even outside the strict subspace defined by the deterministic starting points given by labeled data.

4.3.4 Discussion

Our experiments have shown that with a fair bit of random exploration, we do not find EM initializations that are better than the one from just the labeled data. Of course, with an infinite amount of random exploration, all maxima would be discovered, and the best one identified. Thus, it seems that (1) there are a *lot* of local maxima, which makes it difficult to find the good ones, and (2) the labeled data do a pretty good job of getting us into regions with reasonable local maxima. It's when the labeled data are sparse that things get difficult.

In the previous section we argued that there when labeled data are sparse, there is still significant room for improvement and that improvement could come from finding better initialization parameters for EM. In this section, we have experimented with finding good initializations by random exploration. Even by allocating two orders of magnitude more time for random exploration, we have not found any gains in the quality of our starting points. This suggests that we look elsewhere for improvements.

4.4 Actively Finding Good EM Initializations

Creating labeled data inherently involves human effort. In many real-world domains only a small amount of labeled data can be expected. Typically documents are selected randomly for labeling from all available documents. However it certainly seems reasonable that learning algorithms could perform better if they were able to select which documents get labeled.

In machine learning, the *active learning* setting does exactly this by using a limited amount of interaction with a human labeler. In this setting, the learning algorithm selects which examples to present to the labeler for hand-classification. The task of the learner is to select the most informative examples for labeling, and then learn a classifier from these examples. Typically, active learning is used with algorithms that learn only from labeled data. But the active learning framework by its nature has access to many unlabeled documents (since it must have a selection of documents to choose from). It is then natural to think of applying active learning to learning with labeled and unlabeled data; all the examples that were not selected for labeling provide valuable information if incorporated by algorithms that explicitly use them.

The key idea of this section is that we can use an active learning algorithm to carefully select a few unlabeled documents for labeling. We will use these informative labeled documents to provide high-quality initial values for our model parameters. From this, EM should find high-accuracy parameters using the labeled documents and all the remaining unlabeled documents. We expect this approach to work better than the traditional method of randomly selecting documents for labeling.

4.4.1 Query-by-Committee Active Learning

Active learning aims to select the most informative example—in many settings defined as the one that, if its class label was known, would maximally reduce the error of the classifier trained with that extra example. If one assumes the learner is unbiased then reducing classification error is equivalent to reducing classification variance over the data distribution. This follows from the decomposition of error into bias and variance (Geman et al., 1992). In some cases the expected variance reduction can be estimated empirically, and data can be iteratively selected for labeling by this approach (Cohn et al., 1996).

Frequently, calculating this expected variance reduction in closed-form is prohibitively complex and impractical at best. In these cases active learning can proceed by appealing to the *Query-by-Committee* (QBC) framework (Freund et al., 1997). Here, instead of selecting a document that maximally reduces classification variance, QBC selects a document for labeling that has high classification variance itself. In a consistent error-free learning framework, getting a label for a document with high classification variance eliminates all hypotheses that do not agree with the label. In these cases QBC provides exponential speed-ups in the learning rate over random selection of documents to label (Freund et al., 1997). When the learning task is not so theoretically clean, the intuition behind QBC is that documents with high classification variance lie in regions where the learning algorithm needs help. By getting a true label in that region, significant uncertainty can be eliminated. This approach has been successfully applied to such real-world text tasks as part-of-speech tagging with a HMM representation (Argamon-Engelson & Dagan, 1999) and text classification with Winnow and Perceptron learners (Liere, 1999).

Query-by-Committee gets its name from how it measures the classification variance of an example. It does so by creating a committee of several classifier variants suggested by the data labeled so far. QBC then classifies unlabeled documents with each committee member, and measures the disagreement between their classifications—thus approximating the classification variance. QBC asks for a class label of a document on which the committee disagrees strongly. The newly labeled document is then included in the training data, and a new committee is sampled for making the next set of requests.

With a probabilistic framework for classification the labeled training data specify a posterior distribution over classifiers. Thus, selecting committee members is the

- **Inputs:** Collections $\mathcal{D}^l$ of labeled documents and $\mathcal{D}^u$ of unlabeled documents.
- Calculate the density for each unlabeled document (Eq. 4.6).
- Loop while person willing to label:
    - Loop k times, once for each committee member:
        + Create an initial committee member θ_m by sampling from the Dirichlet distribution defined by the labeled training data (Equation 4.1).
        + *Use θ_m as the initialization for EM and combine the labeled and unlabeled data to find a more likely θ_m (Table 2.1).*
        + Use θ_m to probabilistically label all unlabeled documents (Eq. 2.8).
    - Calculate the disagreement for each unlabeled document (Eq. 4.4), multiply by its density, and request the class label for the one with the highest score.
- Run EM to combine the labeled and the remaining unlabeled data, creating a classifier (Table 2.1).
- **Output:** A classifier, $\hat{\theta}$, that takes an unlabeled document and predicts a class label.

Table 4.2: Our active learning algorithm for finding EM initializations. The step in italics is optional.

same as sampling from this posterior distribution over classifiers. This approach was used successfully by Argamon-Engelson and Dagan (1999). With our probabilistic generative model this approach will be applicable here. When no such framework exists, more ad hoc approaches are taken, such as different random initializations (Liere, 1999).

4.4.2 QBC for Text Classification

This section details how to apply QBC active learning for finding good initializations for EM when learning from labeled and unlabeled data. To do this we need to specify three things: (1) how to form committee members, (2) how to measure committee disagreement, and (3) how to select a document using the disagreement metric. The resulting algorithm is summarized in Table 4.2.

We form a committee of size k to approximate the distribution of classifiers indicated by the labeled data in two different ways. Individual committee members are denoted by m. As discussed in Section 4.3.1, the distribution over classifiers induced by labeled data is a set of Dirichlet distributions, one for each class mixture component. Thus, for each committee member, we draw a parameterization from this set of Dirichlets. In the first approach, "committees of initializations," these parameters are the committee member. Thus the committee approximates the distribution of EM initializations and the direct goal is to select a document to improve the initialization.

In the second approach, "committees of maxima," we use the draws from the Dirichlets as initialization for EM, and incorporate the unlabeled data to change the parameters. After EM converges, this classifier becomes a committee member. Here, we approximate instead the parameter distribution over the corresponding local maxima, and try to select a document that will improve the initialization, in the sense that it puts the initialization in a region with high local maxima. Note that for both these techniques we set the Dirichlets using only the labeled data, and not any unlabeled data. Previously, in Section 4.3, we also used unlabeled data to set the Dirichlets because we wanted our initializations to capture some information about the class-unconditional word frequencies. Here we do not take that approach, as we want to focus on the explicit weaknesses of the labeled data, and not cover them up.

For measuring committee disagreement, Dagan and Engelson (1995) suggest the use of *vote entropy*—the entropy of the class label distribution resulting from having each committee member vote with probability mass $1/k$ for its winning class. One disadvantage of vote entropy is that it does not consider the confidence of the committee members' classifications, as indicated by the class probabilities $\mathrm{P}_m(c_j|d_i;\hat{\theta})$ from each member.

As an improvement, we measure committee disagreement for each document using *Jensen-Shannon divergence* (Lin, 1991). Unlike vote entropy, which compares only the committee members' top ranked class, Jensen-Shannon divergence measures the strength of the certainty of disagreement by calculating differences in the committee members' class distributions, $\mathrm{P}_m(C|d_i)$.[3] Each committee member produces a posterior class distribution, $\mathrm{P}_m(C|d_i)$, where C is a random variable over classes. Jensen-Shannon divergence is an average of the Kullback-Leibler divergence between

[3] While naive Bayes is not an accurate probability estimator (Domingos & Pazzani, 1997), naive Bayes classification scores are somewhat correlated to confidence; the fact that naive Bayes scores can be successfully used to make accuracy/coverage trade-offs (Craven & Slattery, 2001) is testament to this.

each distribution and the mean of all the distributions:

$$\frac{1}{k}\sum_{m=1}^{k} D\left(\mathrm{P}_m(C|d_i)||\mathrm{P}_{avg}(C|d_i)\right), \tag{4.3}$$

where $\mathrm{P}_{avg}(C|d_i)$ is the class distribution mean over all committee members, m: $\mathrm{P}_{avg}(C|d_i) = (\sum_m \mathrm{P}_m(C|d_i))/k$.

Kullback-Leibler divergence, $D(\cdot||\cdot)$, is an information-theoretic measure of the difference between two distributions, capturing the number of extra "bits of information" required to send messages sampled from the first distribution using a code that is optimal for the second. The KL divergence between distributions $\mathrm{P}_1(C)$ and $\mathrm{P}_2(C)$ is:

$$D(\mathrm{P}_1(C)||\mathrm{P}_2(C)) = \sum_{j=1}^{|\mathcal{C}|} \mathrm{P}_1(c_j)\log\left(\frac{\mathrm{P}_1(c_j)}{\mathrm{P}_2(c_j)}\right). \tag{4.4}$$

After disagreement has been calculated by our metric, a document must be selected for a class label request. We consider three ways of selecting documents: stream-based, pool-based, and density-weighted pool-based. Some previous applications of QBC (Dagan & Engelson, 1995; Liere & Tadepalli, 1997) use a simulated stream of unlabeled documents. One at a time a document is considered, measuring classification disagreement among the committee members, and deciding, based on the disagreement, whether to select that document for labeling. Dagan and Engelson (1995) do this heuristically by dividing the vote entropy by the maximum possible entropy to create a probability of selecting the document. Disadvantages of using *stream-based sampling* are that it only sparsely samples the full distribution of possible document labeling requests, and that the decision to label is made on each document individually, irrespective of the alternatives.

An alternative that aims to address these problems is *pool-based sampling.* It selects from among *all* the unlabeled documents the one with the largest disagreement. However, this loses one benefit of stream-based sampling—the implicit modeling of the data distribution—and it may select documents that have high disagreement, but are in unimportant, sparsely populated regions.

We can retain this distributional information by selecting documents using both the classification disagreement and the density of the region around a document. This *density-weighted pool-based sampling* method prefers documents with high classification variance that are also similar to many other documents. The stream approach approximates this implicitly; we accomplish this more accurately (especially when labeling a small number of documents) by modeling the density explicitly.

We approximate the density in a region around a particular document by measuring the average distance from that document to all other documents. Distance, Y, between individual documents is measured by using exponentiated KL divergence:

$$Y(d_i, d_h) = e^{-\beta\, D(\mathrm{P}(W|d_h) \,||\, (\lambda \mathrm{P}(W|d_i) + (1-\lambda)\mathrm{P}(W)))}, \tag{4.5}$$

where W is a random variable over words in the vocabulary; $\mathrm{P}(W|d_i)$ is the maximum likelihood estimate of words sampled from document d_i, (*i.e.*, $\mathrm{P}(w_t|d_i) = N(w_t, d_i)/|d_i|$); $\mathrm{P}(W)$ is the marginal distribution over words; λ is a parameter that determines how much smoothing to use on the encoding distribution (we must ensure no zeroes here to prevent infinite distances); and β is a parameter that determines the sharpness of the distance metric.

In essence, the average KL divergence between a document, d_i, and all other documents measures the degree of overlap between d_i and all other documents; exponentiation converts this information-theoretic number of "bits of information" into a scalar distance.

When calculating the average distance from d_i to all other documents it is much more computationally efficient to calculate the geometric mean than the arithmetic mean, because the distance to all documents that share no words words with d_i can be calculated in advance, and we only need make corrections for the words that appear in d_i. Using a geometric mean, we define density, Z of document d_i to be

$$Z(d_i) = e^{\frac{1}{|\mathcal{D}|} \sum_{d_h \in \mathcal{D}} \ln(Y(d_i, d_h))}. \tag{4.6}$$

We combine this density metric with disagreement by selecting for labeling the document that has the largest product of density (Equation 4.6) and disagreement (Equation 4.3). This density-weighted pool-based sampling selects the document that is representative of many other documents, and about which there is confident committee disagreement.

We have presented algorithmic choices for QBC along several dimensions. We can either form committees of initializations or committees of maxima. We can select documents by simulating a stream, picking straight from the unlabeled pool, or using density weighting when picking from the pool. In the following sections we experimentally evaluate these choices and empirically show that active learning finds initializations for EM that give higher-accuracy classifiers.

4.4.3 Dataset and Protocol

We again use the News5 data, described in Section 4.3.2. On each experimental trial, 20% of the documents are randomly selected for placement in the test set. Initially only one randomly-selected document per class is given as labeled; the remaining documents are unlabeled. Learning proceeds as described in Table 4.2. Experiments are run for 200 active learning iterations, each round selecting one document for labeling. Smoothing parameter λ is 0.5; sharpness parameter β is 3. We made little effort to tune β and none to tune λ. For QBC we use a committee size of three (k=3); our initial experiments showed that committee size had little effect. All EM runs perform seven EM iterations; we never found classification accuracy to improve beyond the seventh iteration. All results presented are averages of ten runs per condition.

4.4.4 Experimental Results

We begin by evaluating selection strategies for QBC. We compare (1) stream-based sampling, (2) pool-based sampling, and (3) density-weighted pool-based sampling. We also calculate the baseline of random selection of documents. For each of these techniques we draw committee members directly from the Dirichlet, and do not run EM to get their associated local maxima. Figure 4.7 shows the quality of the EM initialization in each of these cases, as measured by its classification accuracy.

The best selection technique, density-weighted pool-based sampling achieves 51% accuracy after acquiring only 30 labeled documents. To reach the same accuracy, unweighted pool-based sampling needs 40 labeled documents. If we switch to stream-based sampling we need 51 labelings for 51% accuracy. Our random selection baseline requires 59 labeled documents. Density-weighted pool-based sampling is statistically significantly better than each of the other methods ($p < 0.005$ for each pairing). It is interesting to note that the first several documents selected by this approach are usually FAQs for the various newsgroups. This is clearly an effect of the density weighting as the content of a FAQ is representative of a whole class of documents. Incorporating density-weighting also biases selection towards longer documents, since

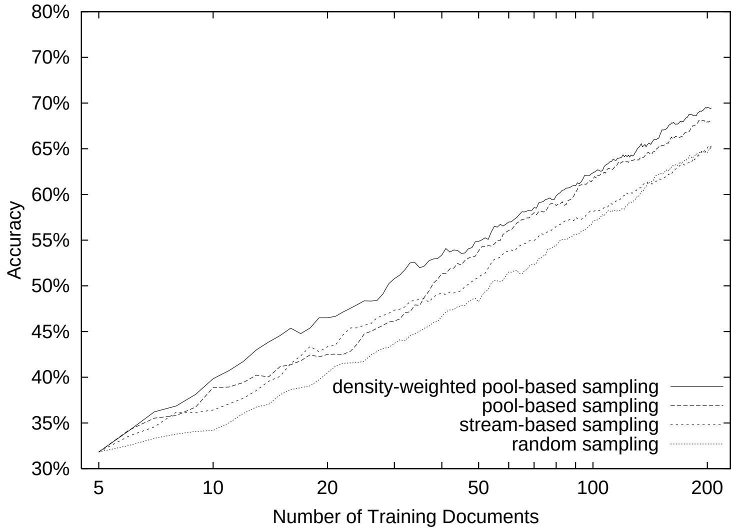

Figure 4.7: A comparison of selection strategies for QBC shows that density-weighted pool-based sampling gives higher-accuracy EM initializations than other strategies. Note that the order of the legend matches the order of the curves and that for resolution the vertical axes do not range from 0 to 100.

these documents have word distributions that are more representative of the corpus as a whole. It is generally better to label long rather than short documents because for the same labeling effort a long document provides information about more words.

Now we compare the two committee creation techniques. For this we use density-weighted pool-based sampling, as this provided the best initializations. Figure 4.8 shows the accuracy of the initializations and the accuracy of the final classifier after EM, compared to the random-selection baseline. Starting with the 30 labeling mark again, committees of initializations reaches 64% accuracy after EM incorporates the unlabeled documents. Committees of local maxima lag only slightly, requiring 32 labeled documents for 64% accuracy. Random selection needs 51 labeled documents. The two committee selection techniques are not statistically significantly different (p = 0.71 N.S.); these two are each statistically significantly better than random selection at this threshold ($p < 0.05$).

Interestingly, although both committee selection methods perform roughly equally after EM incorporates the unlabeled documents, committees of initializations provide more accurate starting points for EM than committees of local maxima. We can understand this by remembering that the two committees focus their attentions differently. Committees of initializations consider specifically the accuracy of the EM

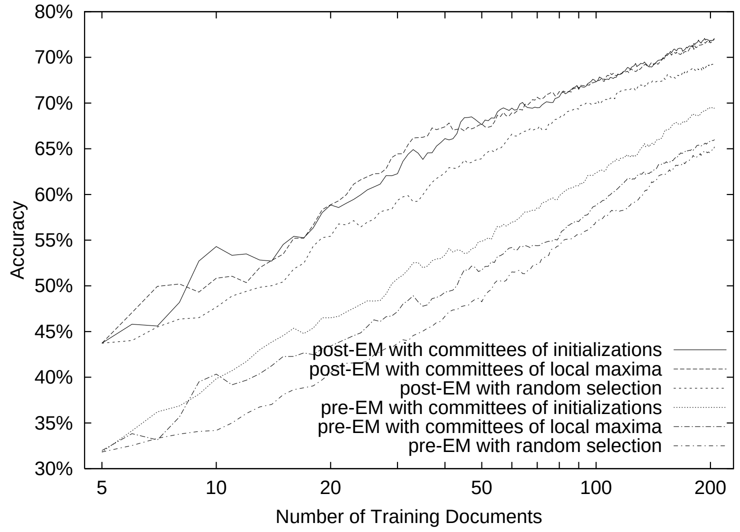

Figure 4.8: The performance of using active learning to select the EM initialization. The two committee formation techniques perform about the same. Both are better than random selection. Note that the order of the legend matches the order of the curves and that for resolution the vertical axes do not range from 0 to 100.

starting point. This indirectly improves the final classification because the quality of the EM starting point is correlated with accuracy of the final classifier. Committees of local maxima focus on the final accuracy, and not the initializations. Thus it is not surprising that this technique's initialization accuracy is lower. What is interesting is that this technique recovers this loss during EM.

4.4.5 Discussion

Other studies have used QBC with probabilistic classifiers in a similar way, and several studies have used active learning to improve text classification. A survey of active learning is given in Section 5.2.4. No previous work has used active learning in combination with algorithms using unlabeled data.

In comparison to previous active learning studies in text classification domains (Lewis & Gale, 1994; Liere & Tadepalli, 1997), the magnitude of our classification accuracy increase is relatively modest. Both of these previous studies consider binary classifiers with skewed distributions in which the positive class is very rare. With a very infrequent positive class, random selection should perform extremely poorly because nearly all documents selected for labeling will be from the negative class. In tasks where the class distributions are more even, random selection should perform much better—making the improvement of active learning less dramatic. In separate work (McCallum & Nigam, 1998b), we experiment with our selection technique on the Reuters domain, which has these skewed prior distributions, and show much larger improvements there. We conclude that our accuracy improvements are good, given that with unskewed class priors, Random selection provides a relatively strong performance baseline.

The results of this section indicate that by actively selecting the labeled examples, we can increase the accuracy of the EM initial parameterization, and thus increase the accuracy of the final classifier. The use of active learning is especially important when only a small number of examples will be labeled; that is when performance with random selection is weakest.

4.5 Avoiding Local Maxima with Deterministic Annealing

In the previous sections of this chapter, we have taken the approach of finding better EM initializations. Through this, with modest success, we have been improving our use of unlabeled data and finding more accurate classifiers. However, local maxima are still a significant problem for EM, especially when labeled training data are sparse. Perhaps it is time to seek alternatives to EM for finding highly probable models. In this section we use a different maximization technique that is more robust to local maxima, deterministic annealing.

Typically variants of, or alternatives to, EM are created for the purpose of speeding up the rate of convergence (McLachlan & Krishnan, 1997, Chapter 4). In the domain of text classification however, we have seen that convergence is very fast. Thus, we can easily consider alternatives to EM that improve the local maxima situation at the expense of slower convergence. Deterministic annealing makes exactly this tradeoff.

The intuition behind deterministic annealing is that it begins by maximizing on a very smooth, convex surface that is only remotely related to our true probability surface of interest. Initially we can find the global maximum of this simple surface. Ever-so-slowly, we change the surface to become both more bumpy, and more close to the true probability surface. If we follow the original maximum as the surface

- **Inputs:** Collections $\mathcal{D}^l$ of labeled documents and $\mathcal{D}^u$ of unlabeled documents.
- Initialize β to a small number near zero.
- Build an initial naive Bayes classifier, $\hat{\theta}$, from the unlabeled documents $\mathcal{D}^u$ spread evenly over classes and the labeled documents $\mathcal{D}^l$. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \arg\max_\theta \mathrm{P}(\mathcal{D}|\theta)\mathrm{P}(\theta)$ (see Equations 2.6 and 2.7).
- Loop while $\beta < 1$:
    - Loop until convergence:
        - **(E-step)** Using the current classifier, $\hat{\theta}$, calculate the expected value of the cluster membership of each unlabeled document, $\hat{z}_{ij}$ (see Equation 4.14).
        - **(M-step)** Re-estimate the clustering parameters, $\hat{\theta}$, given the expected cluster membership of each document. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \arg\max_\theta \mathrm{P}(\mathcal{D}|\theta)\mathrm{P}(\theta)$ (see Equations 2.6 and 2.7).
    - Increase β.
- **Output:** A classifier, $\hat{\theta}$, that takes an unlabeled document and predicts a class label.

Table 4.3: The Deterministic Annealing algorithm described in Section 4.5.1.

gets more complex, then when the original surface is given, we'll still have a highly probable maximum. In this way, it avoids many of the local maxima that EM would otherwise get caught in.

4.5.1 Deterministic Annealing

In this section we sketch the derivation of deterministic annealing as it specifically applies to learning with labeled and unlabeled data for text classification. A more general treatment of deterministic annealing is given by Rose et al. (1992). Where otherwise not mentioned we use the notation introduced in Chapter 2. The deterministic annealing algorithm is summarized in Table 4.3.

Let's approach the problem of learning a classifier with labeled and unlabeled data as one of semi-supervised clustering. The cluster assignments are given for the labeled data and unknown for the unlabeled data. Each cluster is parameterized by a multinomial distribution. When our semi-supervised clustering is complete, each multinomial distribution will correspond to a class and we will have a naive Bayes classifier. We represent the clustering assignments made as the matrix of binary indicator variables $\mathbf{z}$, $\mathbf{z}_i = \langle z_{i1}, \ldots, z_{i|\mathcal{C}|}\rangle$, where $z_{ij} = 1$ iff $y_i = c_j$ else $z_{ij} = 0$; each non-zero entry gives the cluster membership of a datapoint.

We treat semi-supervised clustering as an optimization problem. Our loss function for this optimization is a function of both the model parameters and the class assignments:

$$E(\theta, \mathbf{z}|\mathcal{D}) = -\sum_{d_i \in \mathcal{D}} \sum_{c_j \in \mathcal{C}} z_{ij} \log(\mathrm{P}(c_j|\theta)\mathrm{P}(d_i|c_j;\theta)). \tag{4.7}$$

This loss function is the negative complete data log probability (Section 2.5.1) when given a mixture of multinomials model and deterministic class labels. As we have seen through this thesis, minimizing this loss function is a good target for building a classifier. Although our loss function is motivated by model probability we do not explicitly assume our data were generated by our target parameterization. We think of the parameters as a classifier instead of a model of data generation.

Finding a model for a fixed loss

As a way of finding a model and class assignments that minimize loss let us first solve a related task. The task is to find and select a clustering that has a specific given loss value. The next subsection will give an algorithm for minimizing loss that uses this first step as a sub-component.

For a fixed value E of the loss function there will be a set of pairs of models and cluster assignments that have this loss. If we are given a target loss, and asked to select only one, how would we know which of these pairs to choose or which would be the most likely? To answer this question we apply the principle of maximum entropy (Jaynes, 1957; Good, 1963; Csiszár, 1996). This principle says that in the absence of knowledge, an estimated probability distribution should be as uniform as possible—have maximal entropy. This principle has been commonly and successfully applied for many learning tasks. If we can find the maxent distribution over model parameters given our data we can select the most likely model parameters as our solution. Thus when given a target loss, our goal is to find the most likely parameters for the maxent distribution over parameters and assignments that give the target loss.

This target distribution over model parameters and cluster assignments is the one that maximizes the entropy[4], H:

$$H = -\sum_{\theta,\mathbf{z}} \mathrm{P}(\theta,\mathbf{z}) \log \mathrm{P}(\theta,\mathbf{z}) \tag{4.8}$$

subject to the constraints:

$$\begin{aligned} \sum_{\theta,\mathbf{z}} \mathrm{P}(\theta,\mathbf{z}) &= 1 && (4.9)\\ -\sum_{d_i \in \mathcal{D}} \sum_{c_j \in \mathcal{C}} z_{ij} \log(\mathrm{P}(c_j|\theta)\mathrm{P}(d_i|c_j;\theta)) &= E && (4.10) \end{aligned}$$

The first constraint enforces the requirement that the target distribution must be a valid probability distribution. The second constraint enforces our fixed value for the loss on the model parameters and cluster labels.

By solving this constrained maximization problem using Lagrange multipliers we find the solution to our maxent problem has a Gibbs distribution:

$$\mathrm{P}(\theta,\mathbf{z}|\mathcal{D}) = \frac{\exp(\beta \sum_{d_i \in \mathcal{D}} \sum_{c_j \in \mathcal{C}} z_{ij} \log(\mathrm{P}(c_j|\theta)\mathrm{P}(d_i|c_j;\theta)))}{\int \exp(\beta \sum_{d_i \in \mathcal{D}} \sum_{c_j \in \mathcal{C}} z_{ij} \log(\mathrm{P}(c_j|\theta')\mathrm{P}(d_i|c_j;\theta'))) d\theta'}, \tag{4.11}$$

where β is a Lagrange multiplier determined by the value of E, the desired loss. β can range between 0 (when the desired loss is very large) and ∞ (when the desired loss is very small). Given this form for the *joint* likelihood of models and cluster assignments, how would we select a single classification model? Remember that the cluster assignments for the unlabeled data are only incidental for our purposes of finding a classifier. We can express the probability of each classifier parameterization

[4] Looking ahead, we will want to incorporate priors into the parameters to avoid word probabilities of zero. This is easily done by replacing maximum entropy as our criteria with minimum relative entropy to the prior distribution. The generalization of this derivation to minimum relative entropy is straightforward, but notationally complex. For simplicity, we present the maxent derivation, but give and use the minimum relative entropy algorithm.

independent of the labelings for the unlabeled data by marginalizing Equation 4.11 over $\mathbf{z}$:

$$\mathrm{P}(\theta|\mathcal{D}) = \sum_{\mathbf{z}} \mathrm{P}(\theta, \mathbf{z}|\mathcal{D}) \tag{4.12}$$

By substituting, simplifying and taking the logarithm, we get an expression for log-likelihood of a classification model:

$$\begin{aligned} l(\theta|\mathcal{D}) &= \sum_{d_i \in \mathcal{D}^u} \log \sum_{c_j \in \mathcal{C}} [\mathrm{P}(c_j|\theta)\mathrm{P}(d_i|c_j;\theta)]^{\beta} \\ &\quad + \sum_{d_i \in \mathcal{D}^l} \log([\mathrm{P}(y_i = c_j|\theta)\mathrm{P}(d_i|y_i = c_j;\theta)]^{\beta}). \end{aligned} \tag{4.13}$$

Note that with the exception of the β's, this equation and Equation 2.10 (the incomplete model probability under generative assumptions) are identical. When $\beta = 1$, they are identical. As before, it is computationally intractable to find the most likely model because of the log of sums for the unlabeled data in Equation 4.13. Analogously to Section 2.5 we can apply the EM framework to find a local maximum likelihood solution by iterating the following steps:

- E-step: Calculate the expected value of the class assignments,

$$\hat{z}_{ij}^{(k+1)} = E[z_{ij}] = \frac{[\mathrm{P}(c_j|\hat{\theta}^{(k)})\mathrm{P}(d_i|c_j;\hat{\theta}^{(k)})]^{\beta}}{\sum_{c_r \in \mathcal{C}} [\mathrm{P}(c_r|\hat{\theta}^{(k)})\mathrm{P}(d_i|c_r;\hat{\theta}^{(k)})]^{\beta}}. \tag{4.14}$$

- M-step: Find the most likely model using the expected class assignments,

$$\hat{\theta}^{(k+1)} = \arg\max_{\theta} \mathrm{P}(\theta|\mathcal{D};\hat{\mathbf{z}}^{(k+1)}). \tag{4.15}$$

The M-step is identical to that of Section 2.5, while the E-step includes reference to the loss constraint through β. Note there is a pedagogical distinction to be made between the E-step of Section 2.5 and the EM process here. There we were calculating the expectation of the class labels with respect to the assumed generative distribution of documents. Here we calculate the expectation of the class labels with respect to the maximum entropy distribution over all class labelings and classifiers that have our target loss.

Finding a low-loss model

Given that we can find a (local) maximum likelihood (or maximum a posteriori) model for a fixed loss, we would like to find the model with the minimum loss. Consider how the log-probability of the models (Equation 4.13) is affected by different target losses. When the target loss is very large, β will be very close to zero; the probability of each model will very nearly be its prior probability as the influence of the data will be negligible. In the limit as β goes to zero, the probability surface will be convex with a single global maximum. For a somewhat smaller loss target, β will be small but not negligible. Here, the probability of the data will have a stronger influence. There will no longer be a single global maximum, but several. When $\beta = 1$ we have our familiar probability surface of the previous chapters, with many local maxima.

These observations suggest an annealing-like process for finding a low-loss model. Let the *temperature* of our algorithm be the inverse of β. If we initialize our temperature to be very high, we can easily find the global maximum a posteriori solution with EM, as the surface is convex. When we lower the temperature the probability surface will get slightly more bumpy and complex, as the data likelihood will have a larger impact on the probability of the model. Although more complex, the new maximum will be very close to the old maximum if we have lowered the temperature only slightly. Thus, when searching for the maximum with EM, we can initialize it with the old maximum and will converge to a good maximum for the new probability surface. In this way, we can gradually lower the temperature of our system, all the while tracking a highly probable solution. Eventually, when the temperature (and β) becomes 1, we will have a good local maximum for the probability of the model given our maximum entropy (minimum relative entropy) and fixed loss assumptions. Conveniently, this will also be a good local maximum for our generative model assumptions, as the probability surfaces are identical at this point. Thus, we will have found a high-probability local maximum from labeled and unlabeled data that we can then use for classification.

Note that the computational cost of deterministic annealing is significantly higher than EM. While each iteration takes the same computation, there are many more iterations with deterministic annealing, as the temperature is reduced very slowly. For example, in our experiments, we performed 390 iterations for deterministic annealing, and only seven for EM. When this extra computation can be afforded, the benefit is more accurate classifiers.

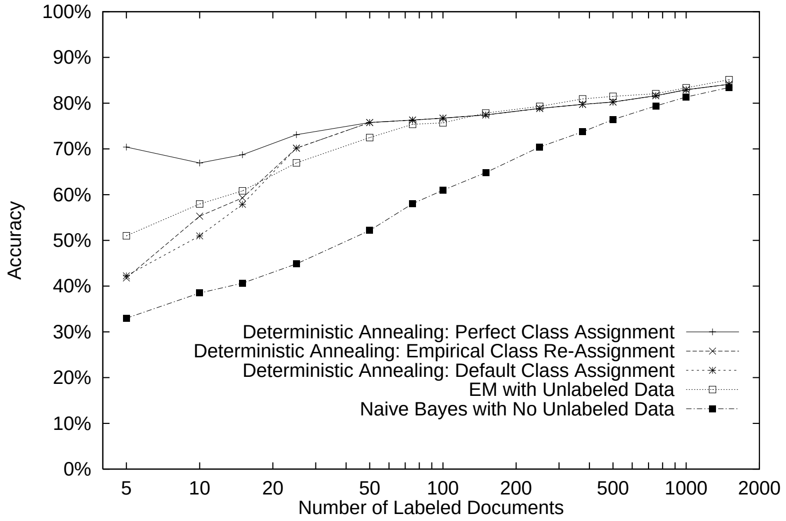

Figure 4.9: The performance of deterministic annealing compared to EM. If class-to-component assignment was done perfectly deterministic annealing would be considerably more accurate than EM when labeled data are sparse. Although the default correspondence is poor, this can be corrected with a small amount of domain knowledge.

4.5.2 Experimental Results

In this section we see empirically that deterministic annealing finds more probable parameters and more accurate classifiers than EM when labeled training data are sparse. For the experimental results, we again use the News5 dataset. We use the same setup and protocol as described in Section 4.3.2. For running the deterministic annealing, we initialize β to 0.02, and at each iteration we increase β by a multiplicative factor of 1.01 until $\beta = 1$. We made little effort to tune these parameters. Deterministic annealing proceeds according to Table 4.3. Since each time we increase β the probability surface changes only slightly, we run only one iteration of EM at each temperature setting.

Figure 4.9 compares classification accuracy achieved with deterministic annealing to that achieved by regular EM. The initial results indicate that the two methods perform essentially the same when labeled data are plentiful, but deterministic annealing actually performs worse when labeled data are sparse. For example with two labeled examples per class (10 total) EM gives 58% accuracy where deterministic annealing gives only 51%. A close investigation of the confusion matrices shows that there is a significant detrimental effect of incorrect class-to-component correspondence with deterministic annealing when labeled data are sparse. This makes

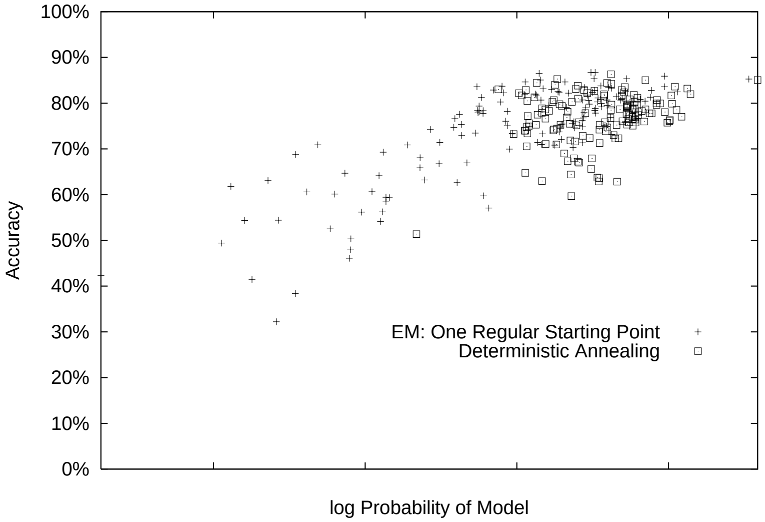

Figure 4.10: A scatterplot comparing the model probabilities and accuracies of EM and deterministic annealing. The results show that deterministic annealing succeeds because it finds models with significantly higher probability.

sense. When the temperature is very high, the global maximum will have each multinomial mixture component very close to its prior. Since the priors are the same, each mixture component will be essentially identical. As the temperature lowers and the mixture components become more distinct, one component can easily track the cluster associated with the wrong class.

In an attempt to remedy this, we alter the class-to-cluster correspondence based on the classification of each labeled example after deterministic annealing is complete. The word counts of each example are subtracted from the final classifier, but there will still be some effect of the example in the cluster. Figure 4.9 shows both the accuracy obtained by empirically selected correspondence, and also the optimal accuracy achieved by perfect correspondence. We see that by empirically setting the correspondence, deterministic annealing improves only marginally. Where before it got 51%, by changing the correspondence we increase this to 55%, still not better than EM at 58%. However if we could perform perfect class correspondence, accuracy with deterministic annealing would be 67%, considerably higher than EM.

To verify that the higher accuracy of deterministic annealing comes from finding more probable models, Figure 4.10 shows a scatterplot of model probability versus accuracy for deterministic annealing (with optimal class assignment) and EM. Two results of note stand out. The first is that indeed deterministic annealing finds much comp.graphics

jpeg	0.0315
image	0.0255
graphics	0.0157
images	0.0122
gif	0.0104
format	0.0059
pub	0.0046
ray	0.0045
tiff	0.0039
siggraph	0.0039

comp.os.ms-windows.misc

windows	0.0287
ei	0.0061
win	0.0056
um	0.0051
dos	0.0050
ms	0.0047
ini	0.0044
microsoft	0.0043
nt	0.0040
el	0.0037

comp.sys.ibm.pc.hardware

scsi	0.0262
ide	0.0234
drive	0.0202
controller	0.0164
bus	0.0106
dx	0.0092
bios	0.0091
drives	0.0085
mb	0.0074
card	0.0070

comp.sys.mac.hardware

apple	0.0227
mac	0.0205
lc	0.0081
duo	0.0079
nubus	0.0069
fpu	0.0068
centris	0.0063
quadra	0.0061
iisi	0.0060
powerbook	0.0056

comp.windows.x

window	0.0174
widget	0.0119
motif	0.0119
xterm	0.0112
server	0.0099
lib	0.0085
entry	0.0081
openwindows	0.0066
usr	0.0060
sun	0.0059

Table 4.4: The top ten words per class of the News5 dataset. The words are sorted by the weighted log-likelihood ratio. Note that from just these ten top words, any person with domain knowledge could correctly correspond clusters and classes.

more probable models, even with a small amount of labeled data. This accounts for the added accuracy of deterministic annealing. A second note of interest is that models found by deterministic annealing still lie along the same probability-accuracy correlation line. This provides further evidence that model probability and accuracy are strongly correlated for this dataset, and that the correlation is not just an artifact of EM.

4.5.3 Discussion

The experimental results show that deterministic annealing indeed could help classification considerably if class-to-component correspondence was a solved problem. Deterministic annealing successfully avoids getting trapped in some poor local maxima and instead finds more probable models. Since these high-probability models are correlated with high-accuracy classifiers, deterministic annealing makes good use of unlabeled data for text classification.

The class-correspondence problem is most severe when there are only limited labeled data. This is because with fewer labeled examples, it is more likely that small perturbations can lead the correspondence astray. However, with just a little bit of human knowledge, the class-correspondence problem can typically be solved trivially. In all but the largest and most confusing classification tasks, it is straightforward to identify a class given its most indicative words, as measured by a metric such as the weighted log-likelihood ratio (Equation 2.14). For example, the top ten words per class of our dataset by this metric are shown in Table 4.4. From just these ten words, any person with even the slightest bit of domain knowledge would have no problem perfectly assigning classes to components. Thus, it is not unreasonable to require just a small amount of human effort to correct the class correspondence after deterministic annealing has finished. Thus, when labeled training data are sparsest, deterministic annealing will successfully find more probable and more accurate models than traditional EM.

If this limited domain knowledge is not available, it should be possible to do the class correspondence automatically. One could perform both EM and deterministic annealing on the data. Since EM solutions generally have the correct class correspondence, this model could be used to fix the correspondence of the deterministic annealing model. That is, we can measure the distance between each EM class multinomial and each deterministic annealing class multinomial (with KL-divergence, for example). Then, we can use this matrix of distances to assign the class labels of the EM multinomials to their closest match to a multinomial in the deterministic annealing model.

Another possible way is to make explicit use of deterministic annealing's insensitivity to the labeled data. Since deterministic annealing is insensitive to its initialization and the influence of the labeled data is minimal, we could perform deterministic annealing using only the unlabeled data. In this way, there would be no residual effects of the labeled data when performing cluster correspondence. This would be exactly the recommendation of Castelli and Cover (1995).

Deterministic annealing was first introduced by Rose et al. (1990) as a way to construct a hierarchy during unsupervised clustering. It was motivated through a strong analogy to statistical physics. The most related deterministic annealing applications are using it to estimate the parameters of a mixture of Gaussians from unlabeled data (Ueda & Nakano, 1995) and constructing a text hierarchy from unlabeled data (Hofmann & Puzicha, 1998).

This chapter has addressed techniques for improving the use of unlabeled data when labeled data are sparse. These conditions provide the best opportunity to benefit from unlabeled data, but also pose some significant challenges. Specifically, our process of integrating unlabeled data through the use of generative models suffers from poor initialization of our EM optimization. We have improved the use of unlabeled data in two different ways. First, if the learning algorithm can interact with a human during their limited labeling effort, it can carefully select which documents get labeled. This helps by finding higher-quality initializations for EM, which lead to more accurate classifiers. We have also used a tempered variation of EM. Deterministic annealing avoids local maxima and finds more probable and more accurate classifiers. With these two different techniques we have made better use of unlabeled data when labeled data are sparse.

Chapter 5

Related Work

The field of text classification is rich with existing and ongoing scientific research. Related theoretical and empirical approaches for incorporating unlabeled data into supervised learning provide a strong foundation for this thesis work. This chapter surveys the current state of these fields and their intersection.

5.1 Text Classification

Text classification has been around in different forms for some time. One of the early application of text classification was to author identification. The seminal work by Mosteller and Wallace (1964) examined authorship of the different Federalist papers using Bayesian analysis of features such as word and sentence length, frequency of function words, and vocabulary diversity. More recently text classification has been applied to a wide variety of practical applications: cataloging news articles (Lewis & Gale, 1994; Joachims, 1998); classifying web pages into a symbolic ontology (Craven et al., 2000); finding a person's homepage (Shavlik & Eliassi-Rad, 1998); automatically learning the reading interests of users (Pazzani et al., 1996; Lang, 1995); automatically threading and filtering email by content (Lewis & Knowles, 1997; Sahami et al., 1998); and book recommendation (Mooney & Roy, 2000).

An early and popular machine learning technique for text classification is naive Bayes (Lewis, 1998; Mitchell, 1997). Its straightforward probabilistic nature has made it amenable to a variety of extensions. Limited word dependencies can be modeled using TAN trees (Sahami, 1996). Leverage of a class hierarchy can be provided through statistical shrinkage (McCallum et al., 1998) or other more ad-hoc techniques (Koller & Sahami, 1997). The one-to-one class-to-component correspondence can be relaxed (Li & Yamanishi, 1997). This thesis has extended naive Bayes through the inclusion of unlabeled data.

There are two different generative models that have been used for naive Bayes. The one used in this thesis is a multinomial (or in language modeling terms, "unigram") model, where the classifier is a mixture of multinomials and tracks the number of times a word appears in a document (McCallum & Nigam, 1998a). This formulation has been used by numerous practitioners of naive Bayes text classification (Lewis & Gale, 1994; Joachims, 1997; Li & Yamanishi, 1997; Mitchell, 1997; McCallum et al., 1998; Lewis, 1998). A second formulation of naive Bayes text classification instead uses a generative model where each word in the vocabulary is a binary feature, and is modeled by a mixture of multi-variate Bernoullis (Robertson & Sparck-Jones, 1976; Lewis, 1992; Larkey & Croft, 1996; Koller & Sahami, 1997). Empirical comparisons show that the multinomial formulation yields classifiers with consistently higher accuracy (McCallum & Nigam, 1998a).

A variety of machine learning techniques other than naive Bayes have been applied to text classification. Support vector machines have recently shown much promise (Joachims, 1998; Dumais et al., 1998). Other approaches have used maximum entropy (Nigam et al., 1999), neural nets (Wiener et al., 1995; Shavlik & Eliassi-Rad, 1998) and several rule learning algorithms (Apte et al., 1994; Cohen & Singer, 1996; Moulinier et al., 1996; Craven et al., 1998). Still others have used memory-based methods like k-nearest neighbor (Yang & Chute, 1994; Cohen & Hirsch, 1998), and a variety of boosting approaches (Schapire & Singer, 2000; Apte et al., 1998; Sebastiani et al., 2000). To date, no single technique has emerged as clearly better than the others, though some recent evidence suggests that kNN and SVMs perform at least as well as other algorithms when there is a lot of labeled data for each class of interest (Yang, 1999).

Most studies into text classification use the simple document representation of bags-of-words, tracking the number of times each word occurs in a document, or even just whether or not it occurred. Consistently, efforts to include more substantial linguistic or semantic information have provided at most modest improvements to classification accuracy. Furnkranz et al. (1998) uses shallow syntactic phrase patterns and finds some improvements to naive Bayes and rule learning algorithms. Mladenic

(1998) selects variable-length phrases for text classification of web pages into the Yahoo hierarchy. Two studies have incorporated into text classification information from WordNet, a semantic network of the English language (Rodriguez et al., 1997; Scott & Matwin, 1998).

5.2 Learning with Labeled and Unlabeled Data

We turn now to a survey of combining labeled and unlabeled data. Much initial work in this area started in statistics, and has recently been joined by the machine learning community.

5.2.1 Likelihood maximization approaches

The idea of learning classifiers from a combination of labeled and unlabeled data is an old one in the statistics community. At least as early as 1968, it was suggested that labeled and unlabeled could be combined for building classifiers with likelihood maximization by testing all possible class assignments (Hartley & Rao, 1968). The seminal paper by Day (1969) presents an iterative EM-like approach for parameters of a mixture of two normals with known covariances from unlabeled data alone. Similar iterative algorithms for building maximum likelihood classifiers from labeled and unlabeled data followed, primarily for mixtures of normal distributions (McLachlan, 1975; Titterington, 1976).

Dempster et al. (1977) presented the theory of the EM framework, bringing together and formalizing many of the commonalities of previously suggested iterative techniques for likelihood maximization with missing data. Its applicability to estimating maximum likelihood (or maximum a posteriori) parameters for mixture models from labeled and unlabeled data (Murray & Titterington, 1978) and then using this for classification (Little, 1977) was recognized immediately. Since then, this approach continues to be used and studied (McLachlan & Ganesalingam, 1982; Ganesalingam, 1989; Shahshahani & Landgrebe, 1994). Three excellent surveys of the history of EM and its application to mixture modeling are the books by McLachlan and Basford (1988), McLachlan and Krishnan (1997) and McLachlan and Peel (2000).

Using likelihood maximization of mixture models for combining labeled and unlabeled data for classification has only recently made its way to the machine learning community (Miller & Uyar, 1996; Nigam et al., 1998; Baluja, 1999). However, other work used this approach earlier for similar purposes. Mixture models have been used as a generative model for unsupervised clustering, whose parameters have been fit with EM (Cheeseman et al., 1988; Cheeseman & Stutz, 1996; Hofmann & Puzicha, 1998). Ghahramani and Jordan (1994) use EM to fill in missing feature values of examples when learning from incomplete data by assuming a mixture model. Hierarchical mixtures-of-experts are similar to mixture models, and their parameters are typically set with EM (Jordan & Jacobs, 1994). The work in this dissertation is the first to explore combining labeled and unlabeled data for text classification.

5.2.2 Discriminative approaches

A transductive support vector machine (Vapnik, 1998) discriminatively finds parameters for a linear separator when given labeled data and the data it will be tested on. In general, this approach is equally applicable to scenarios with labeled and unlabeled data. At a high level, they work by finding the linear separator between the labeled examples of each class that maximizes the margin over both the labeled and unlabeled examples. Joachims (1999) demonstrates the efficacy of this approach for several text classification tasks. Bennett and Demiriz (1999) find small improvements on some UCI datasets with a computationally easier variant of transduction. It seems that the intuition behind transductive SVMs is that they assume decision boundaries lie between classes in low-density regions of instance space, and that the unlabeled examples help find these areas. However, Zhang and Oles (2000) argue both theoretically and experimentally that transductive SVMs are unlikely to be helpful for classification in general.

The maximum entropy discrimination framework (Jaakkola et al., 2000) is another margin-based classification approach that can make use of unlabeled data. It finds the maximum entropy (or minimum relative entropy) distribution over classifier parameters, subject to the (soft) constraints that each labeled example is at least a fixed margin from the decision boundary. Classification is then performed by calculating by integration the expected value of the example's class over the induced parameter distribution. Unlabeled data can be used here by adding a distribution over the unknown class labels that is jointly estimated by maximum entropy. With this approach, margin constraints are also added for the unlabeled data to encourage the estimation to commit to labels for the unlabeled examples. To make this distribution estimation practical, an iterative relaxation algorithm is proposed that converges to a local minima of relative entropy. Encouraging experimental results are given for combining labeled and unlabeled data in the context of predicting DNA splice points.

Recent work by Szummer and Jaakkola (2001) uses unlabeled data in classification through kernel expansion. In essence, the features of each labeled datapoint include the kernel densities between it and all other examples, labeled and unlabeled. Using these features, either the maximum entropy discrimination framework or maximum likelihood derives a linear separator for the classification task. The intuition is that the unlabeled data are used to weight the relative importance of the labeled data based on the instance density. The weighting informs the discriminative training about which examples are more important, and which are less so. In this approach it is essential to carefully select the form and width of the kernel to accurately model the underlying instance distribution. How this should best be done is an open and challenging question.

5.2.3 Theoretical value of unlabeled data

There has been some effort to quantify the relative value of labeled and unlabeled examples when learning a mixture distribution for classification. Unsurprisingly, most results in the literature concern mixtures of two Gaussians. Ganesalingam and McLachlan (1978) calculate a first-order approximation to the asymptotic relative efficiency of labeled and unlabeled examples for classification with univariate normals with known and equal variances. O'Neill (1978) goes one step further and quantifies the asymptotic relative efficiency for multivariate normals with equivalent and known covariance matrices. Ratsaby and Venkatesh (1995) perform a similar analysis, but within the PAC framework. For more general cases beyond mixtures of Gaussians, less is known. For a class of mixture distributions (including normals), Chen (1995) bounds the rate of convergence of parameter estimates from unlabeled examples where the number of mixture components is bounded but not known. These results cover only parameter estimation and not classification error. All of these mentioned results assume (1) the global maximum likelihood parameterization can be found without any problems of local maxima, and (2) the data were actually generated by the model used. For the more general and challenging cases applicable to this thesis, there are no known results.

In an often-cited result, Castelli and Cover (1996) show that labeled examples are exponentially more valuable than unlabeled examples. These results apply only when estimating the class probability parameters where the underlying component distributions are known and correct. In a related result, they show that labeled data reduce error exponentially fast with an infinite amount of unlabeled data when the component distributions are known, but the class-to-component correspondence is not (Castelli & Cover, 1995). In a recent study, Zhang and Oles (2000) examine the value of unlabeled data for discriminative trainers such as transductive SVMs and also for active learning. As mentioned above, they question the generality of the helpfulness of transductive SVMs.

5.2.4 Active Learning

Although often not explicitly stated, selective sampling is another way of integrating unlabeled data into supervised learning. Selective sampling (Cohn et al., 1994) is a form of active learning where one must select an existing example for labeling. One approach for selecting examples is to try to maximally reduce the variance component of classification error (Cohn et al., 1996). Another approach, the one taken in this thesis, is that of query-by-committee (Seung et al., 1992; Freund et al., 1997), whereby a committee of classifier variants is used to select an example that has high classification variance itself. A theoretical analysis shows that in a consistent error-free learning domain, QBC can exponentially reduce the number of labeled examples needed for learning (Freund et al., 1997). QBC has also shown to be a powerful paradigm in practice. Argamon-Engelson and Dagan (1999) use a query-by-committee approach to learn a part-of-speech tagger. Similarly to the work in Section 4.4, they use statistical models of their classifiers to create the committee. In contrast, they use stream-based sampling and vote entropy to measure committee disagreement. For text classification, we have shown that pool-based sampling and a Jensen-Shannon divergence based metric perform better. Liere (1999) uses committees of Winnow or Perceptron learners for QBC active learning for text classification. Here, a document is selected for labeling for which two randomly selected committee members disagree on the class label. Committee members are formed by multiple random initializations of the classifiers.

Several other studies have investigated active learning and selective sampling specifically for the domain of text categorization. Lewis and Gale examine uncertainty sampling and relevance sampling in a pool-based setting (Lewis & Gale, 1994;

Lewis, 1995). These techniques select documents based on a single classifier instead of a committee, and thus do not approximate classification variance. Uncertainty sampling selects for labeling the example that is classified with the most uncertainty (uniform class posteriors), where relevance sampling chooses the example with the most certainty (peaked class posteriors). Schohn and Cohn (2000) use an approach very similar to uncertainty sampling, but for support vector machines. They select for labeling the example that is closest to the linear decision boundary given by the classifier. Tong and Koller (2000) also perform selective sampling with support vector machines, but their approach is motivated by trying to maximally reducing the size of the version space of good hypotheses. All of these studies have shown significant benefit from active learning when applied to text classification domains.

5.2.5 Other uses of unlabeled data for supervised learning

The co-training setting allows unlabeled data to be used in new ways. It specifies that every example is described by two disjoint views onto the data. For example, with a web classification task, each instance has words occurring on a web page, and also words on hyperlinks pointing to the web page. Blum and Mitchell (1998) show that under certain theoretical assumptions, a weak learner can be arbitrarily improved given sufficient unlabeled examples. They also present the co-training algorithm that iteratively selects an unlabeled example, gives it a label, and relearns. Nigam and Ghani (2000) argue that the co-training algorithm and its variants succeed in part because they are more robust to the assumptions of their underlying classifier representations. Collins and Singer (1999) present a boosting-based algorithm, coBoost, for learning in the co-training setting; it tries to minimize the disagreement on the unlabeled data between classifiers that use different views of the data. Goldman and Zhou (2000) show that co-training approaches can succeed on datasets without disjoint views when carefully selected underlying classifiers are used.

Several developments using *distantly labeled* data have proven to be useful. Distantly labeled data are data labeled for a related task, but do not have a direct map to the task at hand. The AutoSlog-TS system (Riloff, 1996) takes documents labeled as in or out of the domain of interest and automatically extracts indicative case frames that often match items to be extracted. Seymore et al. (1999) have used distantly-labeled data to estimate output parameters of an HMM used for information extraction in the research paper domain. In a similar vein, Zelikovitz and Hirsh

(2000) use unlabeled data as background knowledge to augment a nearest-neighbor classifier. Instead of matching a test example directly to its closest labeled example, they instead match a test example to a labeled example by measuring their similarity to a common set of unlabeled examples.

Several bootstrapping techniques allow learning algorithms to begin with nearly no labeled data and iteratively develop a concept of interest. Riloff and Jones (1999) start with just a small dictionary of known locations and a set of unlabeled data to bootstrap a much larger dictionary of locations and case frame patterns indicative of locations. Yarowsky (1995) bootstraps a word sense disambiguation algorithm starting with a small set of seed collocations of words and senses.

Unlabeled data have also been used to reduce or prevent overfitting. For example, there is strong evidence for overfitting when the disagreement on the unlabeled data between two candidate classifiers or regressors is larger than the sum of their errors on labeled data. This observation has been used for selecting the best complexity of a polynomial for regression (Schuurmans, 1997) and for pruning decision trees (Schuurmans & Southey, 2000). Cataltepe and Magdon-Ismail (1998) use unlabeled or test data to reduce overfitting in linear regression by augmenting the minimization criteria of mean squared error with terms based on unlabeled data.

Chapter 6

Conclusions

This dissertation has addressed the problem of integrating unlabeled data into supervised learning for text classification. Labeled data are expensive to collect, as a human must take the time and effort to label it. Thus it is frequently the case that labeled training data are sparse. By contrast unlabeled data are often inexpensive and plentiful. This is especially true for text classification tasks where almost any type of text is readily available in electronic form.

6.1 Findings

There are five significant findings of this dissertation:

Unlabeled data are useful for text classification.

We have taken the approach of first specifying a simple statistical generative model class for documents, and then choosing parameters for the model that are highly probable given the evidence of the labeled and unlabeled documents. Our basic model of the document distribution is a mixture model where each mixture component generates documents for a class with a multinomial distribution over words. With labeled data alone this is the familiar naive Bayes text classifier, where the maximum a posteriori parameters are found with closed form equations. When there is also access to unlabeled data, we can find a local maximum for these parameters given all the data using the Expectation-Maximization (EM) technique. The parameters of this model can be turned around by Bayes' rule and used for classification. In some domains there is a strong correspondence between model probability and classification accuracy; here finding more probable parameters yields more accurate classifiers. In experiments detailed in Section 2.6 we use labeled and unlabeled data to build text classifiers for 20 Newsgroups, a collection of UseNet articles. For this dataset we show that (1) classification accuracy and model probability are strongly correlated, and (2) the use of unlabeled data reduces classification error by 30% when labeled training data are sparse. Thus we conclude that integrating unlabeled data into supervised learning can reduce the error of text classifiers.

Initially it may seem surprising that a generative model approach and EM can increase text classification accuracy using unlabeled data. The intricacies of text documents are not captured by a mixture of multinomials model. What is interesting in this finding is that even though our model is an approximation, maximizing the probability of this model provides increased accuracy in our classifiers. The model captures enough information from the documents for the purposes of classification.

Modeling sub-topic structure makes models more representative.

Some datasets cannot be adequately modeled with the basic generative process. In these cases using unlabeled data for parameter estimation can actually increase classification error by finding parameters that only poorly match the true data distribution. When a class is complex, with several or many sub-topics, a single multinomial mixture component is insufficient; a more representative model is required. In Section 3.2.3 we use multiple mixture components per class for the Reuters dataset to model a single class containing news stories about many topics. Using unlabeled data there with the basic model often increases classification error. However, a generative model with multiple mixture components per class is representative enough for this dataset to bring model probability and classification accuracy into correlation; classification error decreases by 39% across topics compared to using labeled data alone. Thus, with a sufficiently representative model, maximizing the model posterior probability finds more accurate classifiers using labeled and unlabeled data.

With labeled data alone, other researchers have shown that generative models that allow for some class-conditional word dependencies yield more accurate classifiers. When unlabeled data are used, the same principles of model representation also arise, but even more strongly. We assume model correctness when labeling the unlabeled data during parameter estimation; if this assumption is too strongly violated, then the generated class labels will reflect the biases of the model and will not provide useful or accurate model parameters. In these cases it is necessary to adjust the bias for the assumed parametric form of the model more closely matches the true document distribution.

Modeling super-topic hierarchical class relationships reduces overfitting.

Many classification domains have hierarchical relationships among its classes. The basic generative process ignores these relationships when it models each class as an independent mixture component. Therefore, parameters in different classes that are closely related must be re-estimated for each class separately. This increases the opportunity for overfitting the unlabeled data. By explicitly modeling hierarchical class relationships, we make more efficient use of our unlabeled data by estimating these shared parameters only once with more data. This approach reduces overfitting by making the model probability given the unlabeled data more closely match the model probability over the true document distribution. The Cora dataset has a computer science hierarchy that can be leveraged in just this way. In Section 3.3.3 we show that without the hierarchy unlabeled data reduce classification error. With the more representative hierarchical model, we make more efficient use of the data and error is reduced by 33% over a naive Bayes baseline relative to human performance. Thus, we conclude that modeling known hierarchical relations in the generative process can more efficiently use unlabeled data and produce more accurate text classifiers.

Active learning creates improved EM initializations.

In our algorithm for incorporating unlabeled data into supervised learning, the labeled and unlabeled data play different roles. The primary effect of the labeled data is to initialize the maximization process; the unlabeled data have their significant effect during the EM iterations. The quality of the initialization is typically the first-order determinant of the accuracy of the resulting classifier. This provides an opportunity to use unlabeled data to further improve text classification accuracy. With limited interaction with a human labeler, we use Query-by-Committee active learning to select which unlabeled documents should get labeled to provide high-quality initializations for EM. In Section 4.4.4 we apply this approach to the News5 dataset. By initializing

EM from actively selected documents we reduce by 41% the number of labelings needed to reach a given accuracy threshold for our classifier. Thus, we conclude that actively selecting unlabeled documents for labeling reduces the bottleneck imposed by labeled data.

Deterministic annealing finds more accurate text classifiers than EM.

The reason the EM initialization strongly influences classification accuracy is that EM gets trapped in one of the many local maxima in probability space. However, EM is not the only approach for performing this maximization. An alternative technique is deterministic annealing. It is more robust to local maxima because is first maximizes on a very smooth surface and then gradually makes the probability surface more bumpy, maximizing along the way, and ending with a maximum in the original probability surface. Through this cooling-like process, deterministic annealing is able to track a high-probability maximum and settle on a high-accuracy classifier. In Section 4.5.2 we apply this technique to the News5 dataset. Deterministic annealing consistently finds more probable solutions than EM, especially when labeled data are sparse and the initializations are poor. Deterministic annealing is more prone to errors in class-to-component correspondence, but these can be easily corrected with a minimum of human effort. With its more probable parameters, deterministic annealing gives classifiers that reduce classification error by 21% over EM. Thus we conclude that deterministic annealing finds more probable models (and thus more accurate text classifiers) than EM by more effectively avoiding local maxima.

In summary, we have given a basic approach for using unlabeled data for classification: maximizing the probability of statistical generative models. This approach works well when the model is representative enough so that classification accuracy is correlated to model probability. When the basic model is not representative, a more accurate model can often be found and used successfully. With a representative model, unlabeled data help, but labeled data is still the bottleneck for even better performance. This issue can be addressed by actively selecting the labeled data, or by using maximization techniques that are not so sensitive to the labeled data.

6.2 Future Directions

The work of this thesis suggests several directions of further research that are deserving of attention. We outline a few of them here.

When to use a generative model approach

Other recent approaches for incorporating unlabeled data into supervised learning do not take a generative model approach (Joachims, 1999; Szummer & Jaakkola, 2001; Jaakkola & Haussler, 1999). They take a discriminative approach, in that they focus on directly estimating the decision boundary between classes. One interesting and important area of future work is to understand what types of domains are more amenable to one approach or the other. As an extreme example, if the data really were generated by our statistical model, then a posterior model maximization approach would be better; the bias of the algorithm would be exactly correct and allow the most efficient use of the data. We know, however, that any real-world text dataset will not be completely captured in a statistical model. If a sufficiently representative model for the data can not be found, then it is likely that a discriminative approach would be more appropriate.

The discriminative approaches make assumptions of their own that are subject to violations. For example, transductive SVMs (Joachims, 1999) assume there is a low density region through which the linear separator passes. For some domains, this need not be the case, and a generative approach would be more suited. It would be exciting to understand how much the assumptions of each approach can be violated.

Ideally there would be a test, based on the unlabeled and limited labeled data, that could predict which approach would be more beneficial. For example, there are a number of statistical goodness-of-fit tests for multinomial and mixture of multinomials distributions (e.g., Chen, 1998; Zelterman, 1987; Cressie & Read, 1984). If a probability-based parameterization proved to have a poor fit to a mixture of multinomials distribution, this may suggest that a discriminative approach would be better suited to the domain. However, we hypothesize that in many cases the fit to the model will not be good even when the model is representative enough for beneficial use of generative approaches. Thus, existing tests may not be sufficient.

A better approach might directly compare the relative accuracies of the document distributions as measured by both a mixture of multinomials and a non-parametric kernel-based density estimator. If the non-parametric estimator was a much better predictor of previously-unseen data, kernel-based discriminative approaches (Szummer & Jaakkola, 2001) may perform best.

Another possibility would be to explicitly examine the correlation between the probability of a generative model and the classification accuracy. With a modest but not extremely small set of labeled examples, accuracy can be measured on many different parameter settings found by totally unsupervised clustering. The strength of the correlation may be a good predictor for the appropriateness of a generative model. For example, in Section 4.3.3 we performed many runs of EM with random initializations. On that data, displayed in Figure 4.6, the correlation between model probability and accuracy was high, 0.8915, perhaps indicating the appropriateness of our approach on this dataset.

Feature Selection and Multiple Classification Tasks

Taken to an extreme, the generative model approach suggests there is only a single classification task per domain. For example, there is just one global maximum parameterization for a mixture of five multinomials for the News5 data, and we have set this parameterization as our goal. However, documents are not typically authored with the intent of fitting into arbitrary categories. Additionally, we can imagine several different classification tasks for the same domain. For example, in addition to the given task of newsgroup source, we might be interested in classifying the same data by nationality of author, relevance to computer science researchers, or relevance to system administrators. How might we reconcile the contradiction of using generative models for multiple classification tasks on the same data?

One way to reconcile this is through the use of feature selection. With different sets of features for the domain and generative model, there can be dramatically different maximum a posteriori parameterizations. Feature selection is often an essential step in building a text classifier from labeled data alone (Yang & Pedersen, 1997). However, with only limited labeled data it is difficult to select an appropriate set of features considering that many words have not even occurred in the labeled data.

We hypothesize that there is a new and interesting set of feature selection algorithms that would be appropriate for application to combinations of labeled and unlabeled data. One promising direction should be iterative feature selectors, perhaps in the style of Improved Iterative Scaling (Della Pietra et al., 1997) or Boosting

(Schapire & Singer, 2000). The model maximization and the feature selection could be interleaved, allowing for influence from the unlabeled data while still maintaining the direction given by the labeled data. This may also provide some robustness to overfitting and local maxima in the same way as deterministic annealing.

Appendix A

Complete Hierarchy and Keywords

This appendix presents the entire hierarchy and key words and phrases used for the Cora experiments in Section 3.3.

- Information Retrieval
  - Retrieval
    - information retrieval
  - Extraction
    - information extraction
    - wrapper induction
    - wrapper www
  - Filtering
    - text classification
    - document classification
    - document categorization
    - document filtering
  - Digital Library
    - digital library
- Encryption and Compression
  - Encryption
    - encryption
    - cryptographic
    - cryptology
    - cryptanalyzing
    - cryptanalysis
    - cryptography
    - mutual distrust
    - decentralized authority
    - secure secret
  - Security
    - computer security
    - security hole
    - security holes
    - security attack
    - SSL
    - network security
    - firewall kerberos
  - Compression
    - compression
    - entropy code
    - video audio
    - audio coding
    - video coding
    - MPEG
- Human Computer Interaction
  - Multimedia
    - multimedia
  - Interface Design
    - GUI
    - interface design
  - Graphics and Virtual Reality
    - virtual reality
    - telepresence
    - computer graphics
    - siggraph
  - Cooperative
    - cscw
  - Wearable Computers
    - wearable computers
    - wearable computer

- Artificial Intelligence
    - Expert Systems
        - expert systems
    - Knowledge Representation
        - knowledge representation
    - Machine Learning
        - Reinforcement Learning
            - reinforcement learning
            - temporal difference learning
        - Genetic Algorithms
            - genetic algorithms
            - genetic algorithm
            - genetic programming
            - natural selection evolutionary
            - evolutionary computation
        - Probabilistic Methods
            - bayes rule
            - density estimation
            - bayesian network
            - bayes network
            - bayesian networks
            - bayes networks
            - belief revision
            - uncertain inference
            - probabilistic reasoning
            - uai
        - Neural Networks
            - neural network
            - neural networks
            - gradient descent
            - projection pursuit
            - self-organizing map
            - ijcnn
        - Theory
            - pac
            - colt
            - warmuth freund schapire
        - Rule Learning
            - rule learning
            - ILP
    - Case-Based
        - case based
        - instance based
        - memory based
    - Robotics
        - robotics robot robots
    - Vision and Pattern Recognition
        - pattern recognition
        - pami
        - computer vision
    - Speech
        - speech recognition
    - Planning
        - planning
        - temporal reasoning
        - reasoning time
    - NLP
        - NLP
        - natural language processing
    - Theorem Proving
        - theorem proving
    - Agents
        - agent agents multiagent
        - artificial life
    - Games and Search
        - game search
        - combinatorial optimization
        - stochastic optimization
    - Data Mining
        - data mining
        - database mining
- Databases
    - Temporal
        - database temporal
        - databases temporal
    - Deductive
        - deductive database deductive databases
    - Concurrency
        - concurrency database
    - Query Evaluation
        - query-evaluation database

    - database queries
  - Object Oriented
    - oodbms
    - object oriented database
  - Relational
    - relational database
  - Performance
    - database real-time
    - databases real-time
    - database parallel
    - databases parallel
    - database scalable
- Programming
  - Garbage Collection
    - garbage collection
  - Semantics
    - programming language semantics
    - denotational semantics programming language
  - Compiler Design
    - compiler design
    - compiler algorithm
    - parallelizing compiler
    - compiler language
    - optimized code
    - compiling programs
  - Debugging
    - debugging
  - Java
    - java
  - Object Oriented
    - object oriented programming
    - smalltalk
  - Functional
    - functional programming
    - lisp
  - Logic
    - logic programming
    - logic programming
    - prolog
    - prolog
  - Software Development
    - software engineering
    - design tools
    - software metrics
    - programming environments
    - computer aided engineering
    - software reuse
    - software portability
    - configuration management
- Operating Systems
  - Memory Management
    - memory management
    - shared memory
  - Distributed
    - distributed system
    - distributed computing
    - distributed computing
    - distributed os
    - distributed system
    - distributed systems
    - distributed network environment
    - distributed operating systems
    - distributed operating system
    - distributed file system
    - distributed file systems
    - network file system
    - mobile computing
    - networks of workstations
    - cluster of workstations
    - distributed storage system
  - Realtime
    - realtime
    - real time
    - RTSS
  - Fault Tolerance
    - fault tolerance
    - tolerate faults
- Networking
  - Protocols
    - network protocols

    - communication protocol
    - communication protocols
    - multicast
    - mbone
    - atm
    - tcp/ip
    - udp
  - Routing
    - routing
    - qos
    - switched networks
    - switched network
    - routing networks
  - Internet
    - internet
    - internet architecture
    - web caching
  - Wireless
    - wireless network
    - wireless networking
    - mobile network
- Hardware and Architecture
  - High Performance Computing
    - high performance computing
  - VLSI
    - VLSI
  - Memory Structures
    - memory structures
    - TLB NUMA hardware
  - Distributed Architectures
    - distributed architectures
    - distributed architecture
    - distributed hardware
  - Microprogramming
    - microprogramming
    - microprocessor controller
  - Logic Design
    - circuit
    - circuits
    - CMOS
    - logic design
    - gate level
    - multi-level circuit
    - analog circuits
    - analog systems
    - analog converters
    - gate circuit
    - gate circuits
    - circuit transistor
    - circuit fan
  - Input Output and Storage
    - disk drive
    - SCSI
    - i/o
- Data Structures, Algorithms and Theory
  - Randomized
    - randomized algorithms
  - Parallel
    - parallel algorithms
  - Formal Languages
    - finite state machine method
    - context-free languages
    - formal model automata
    - theory automata
  - Computational Complexity
    - space complete
    - complexity of deciding
    - complexity class
    - bounds complexity
  - Sorting
    - sorting
  - Hashing
    - hashing
  - Computational Geometry
    - computational geometry
  - Logic
    - finite model theory
    - program verification
    - format analysis verification
    - modal logic
  - Quantum Computing
    - quantum

Bibliography

Apte, C., Damerau, F., & Weiss, S. M. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, *12*(3), 233–251.

Apte, C., Damerau, F., & Weiss, S. M. (1998). Text mining with decision trees and decision rules. *Workshop on Learning from Text and the Web, Conference on Automated Learning and Discovery.*

Argamon-Engelson, S., & Dagan, I. (1999). Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, *11*, 335–360.

Baluja, S. (1999). Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled examples. *Advances in Neural Information Processing Systems 11*, pp. 854–860.

Bennett, K., & Demiriz, A. (1999). Semi-supervised support vector machines. *Advances in Neural Information Processing Systems 11*, pp. 368–374.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pp. 92–100.

Boyan, J. A. (1998). *Learning evaluation functions for global optimization.* Doctoral dissertation, Computer Science Department, Carnegie Mellon University.

Carlin, B., & Louis, T. (1996). *Bayes and empirical Bayes methods for data analysis.* Chapman and Hall.

Castelli, V., & Cover, T. M. (1995). On the exponential value of labeled samples. *Pattern Recognition Letters*, *16*(1), 105–111.

Castelli, V., & Cover, T. M. (1996). The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, *42*(6), 2101–2117.

Cataltepe, Z., & Magdon-Ismail, M. (1998). Incorporating test inputs into learning. *Advances in Neural Information Processing Systems 10*, pp. 437–443.

Cheeseman, P., Kelley, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AutoClass: A Bayesian classification system. *Machine Learning: Proceedings of the Fifth International Conference*, pp. 54–64.

Cheeseman, P., & Stutz, J. (1996). Bayesian classification (AutoClass): Theory and results. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy (Eds.), *Advances in knowledge discovery and data mining*. MIT Press.

Chen, J. (1995). Optimal rate of convergence for finite mixture models. *The Annals of Statistics*, *23*(1), 221–233.

Chen, J. (1998). Penalized likelihood-ratio test for finite mixture models with multinomial observations. *The Canadian Journal of Statistics*, *26*(4), 583–599.

Cohen, W. W., & Hirsch, H. (1998). Joins that generalize: text categorization using WHIRL. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 169–173.

Cohen, W. W., & Singer, Y. (1996). Context-sensitive learning methods for text categorization. *SIGIR '96: Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 307–315.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*(2), 201–221.

Cohn, D., Ghahramani, Z., & Jordan, M. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, *4*, 129–145.

Collins, M., & Singer, Y. (1999). Unsupervised models for named entity classification. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. New York: John Wiley and Sons.

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

Craven, M., & Slattery, S. (2001). Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, *43*(1/2), 97–119.

Craven, M., Slattery, S., & Nigam, K. (1998). First-order learning for web mining. *Proceedings of the 10th European Conference on Machine Learning*, pp. 250–255.

Cressie, N., & Read, T. R. C. (1984). Multinomial goodness-of-fit tests. *Journal of the Royal Statistical Society, Series B*, *46*(3), 440–464.

Csiszár, I. (1996). Maxent, mathematics, and information theory. In K. Hanson and R. Silver (Eds.), *Maximum entropy and Bayesian methods*. Kluwer Academic Publishers.

Dagan, I., & Engelson, S. P. (1995). Committee-based sampling for training probabilistic classifiers. *Machine Learning: Proceedings of the Twelfth International Conference*, pp. 150–157.

Day, N. E. (1969). Estimating the components of a mixture of normal distributions. *Biometrika*, *56*(3), 463–474.

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(4), 380–393.

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

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10*(7), 1895–1924.

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, *29*(2/3), 103–130.

Dumais, S. T., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. *Proceedings of the Seventh International Conference on Information and Knowledge Management*, pp. 148–155.

Elworthy, D. (1994). Does Baum-Welch re-estimation help taggers? *Proceedings of the Fourth ACL Conference on Applied Natural Language Processing*, pp. 53–58.

Freund, Y., Seung, H., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, *28*(2/3), 133–168.

Friedman, J. H. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1*(1), 55–77.

Furnkranz, J., Mitchell, T., & Riloff, E. (1998). A case study in using linguistic phrases for text categorization on the WWW. *Learning for Text Categorization: Papers from the AAAI Workshop*, pp. 5–12. Tech. rep. WS-98-05, AAAI Press.

Ganesalingam, S. (1989). Classification and mixture approaches to clustering via maximum likelihood. *Applied Statistics*, *38*(3), 455–466.

Ganesalingam, S., & McLachlan, G. J. (1978). The efficiency of a linear discriminant function based on unclassified initial samples. *Biometrika*, *65*, 658–662.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*(1), 1–58.

Ghahramani, Z., & Jordan, M. I. (1994). Supervised learning from incomplete data via an EM approach. *Advances in Neural Information Processing Systems 6*, pp. 120–127.

Goldman, S., & Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. *Proceedings of the Seventeenth International Conference on Machine Learning: ICML-2000*.

Good, I. J. (1963). Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *The Annals of Mathematical Statistics*, *34*(3), 911–934.

Hartley, H. O., & Rao, J. N. K. (1968). Classification and estimation in analysis of variance problems. *Review of International Statistical Institute*, *36*, 141–147.

Hofmann, T., & Puzicha, J. (1998). *Statistical models for co-occurrence data*. Tech. rep. AI Memo 1625. Artificial Intelligence Laboratory, MIT.

Jaakkola, T., & Haussler, D. (1999). Exploiting generative models in discriminative classifiers. *Advances in Neural Information Processing Systems 11*, pp. 487–493.

Jaakkola, T., Meila, M., & Jebara, T. (2000). Maximum entropy discrimination. *Advances in Neural Information Processing Systems 12*, pp. 470–476.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical Review*, *106*, 620–630.

Joachims, T. (1997). A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. *Machine Learning: Proceedings of the Fourteenth International Conference*, pp. 143–151.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98, Tenth European Conference on Machine Learning*, pp. 137–142.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *Machine Learning: Proceedings of the Sixteenth International Conference*.

Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, *6*(2), 181–214.

Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. *Machine Learning: Proceedings of the Fourteenth International Conference*, pp. 170–178.

Lang, K. (1995). NewsWeeder: Learning to filter netnews. *Machine Learning: Proceedings of the Twelfth International Conference*, pp. 331–339.

Larkey, L. S., & Croft, W. B. (1996). Combining classifiers in text categorization. *SIGIR '96: Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 289–297.

Lewis, D. D. (1992). An evaluation of phrasal and clustered representations on a text categorization task. *SIGIR '92: Proceedings of the Fifteenth Annual International*

ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 37–50.

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

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. *Machine Learning: ECML-98, Tenth European Conference on Machine Learning*, pp. 4–15.

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. *SIGIR '94: Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3–12.

Lewis, D. D., & Knowles, K. A. (1997). Threading electronic mail: A preliminary study. *Information Processing and Management*, *33*(2), 209–217.

Lewis, D. D., & Ringuette, M. (1994). A comparison of two learning algorithms for text categorization. *Third Annual Symposium on Document Analysis and Information Retrieval*, pp. 81–93.

Lewis, D. D., Schapire, R. E., Callan, J. P., & Papka, R. (1996). Training algorithms for linear text classifiers. *SIGIR '96: Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 298–306.

Li, H., & Yamanishi, K. (1997). Document classification using a finite mixture model. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 39–47.

Liere, R. (1999). *Active learning with committees: An approach to efficient learning in text categorization using linear threshold algorithms.* Doctoral dissertation, Department of Computer Science, Oregon State University.

Liere, R., & Tadepalli, P. (1997). Active learning with committees for text categorization. *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pp. 591–596.

Lin, K. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, *37*(1), 145–151.

Little, R. J. A. (1977). Discussion on the paper by Professor Dempster, Professor Laird and Dr. Rubin. *Journal of the Royal Statistical Society, Series B*, *39*(1), 25.

McCallum, A., & Nigam, K. (1998a). A comparison of event models for naive Bayes text classification. *Learning for Text Categorization: Papers from the AAAI Workshop*, pp. 41–48. Tech. rep. WS-98-05, AAAI Press.

McCallum, A., & Nigam, K. (1998b). Employing EM in pool-based active learning for text classification. *Machine Learning: Proceedings of the Fifteenth International Conference*, pp. 350–358.

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of Internet portals with machine learning. *Information Retrieval*, *3*(2), 127–163.

McCallum, A., Rosenfeld, R., Mitchell, T., & Ng, A. (1998). Improving text classification by shrinkage in a hierarchy of classes. *Machine Learning: Proceedings of the Fifteenth International Conference*, pp. 359–367.

McLachlan, G., & Basford, K. (1988). *Mixture models.* New York: Marcel Dekker.

McLachlan, G., & Peel, D. (2000). *Finite mixture models.* New York: John Wiley and Sons.

McLachlan, G. J. (1975). Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association*, *70*(350), 365–369.

McLachlan, G. J., & Ganesalingam, S. (1982). Updating a discriminant function on the basis of unclassified data. *Communications in Statistics: Simulation and Computation*, *11*(6), 753–767.

McLachlan, G. J., & Krishnan, T. (1997). *The EM algorithm and extensions.* New York: John Wiley and Sons.

Merialdo, B. (1994). Tagging English text with a probabilistic model. *Computational Linguistics*, *20*(2), 155–171.

Miller, D. J., & Uyar, H. (1996). A generalized Gaussian mixture classifier with learning based on both labelled and unlabelled data. *Proceedings of the 1996 Conference on Information Science and Systems.*

Mitchell, T. M. (1997). *Machine learning.* New York: McGraw-Hill.

Mladenic, D. (1998). *Machine learning on non-homogeneous, distributed text data.* Doctoral dissertation, Faculty of Computer and Information Science, University of Ljubljana, Slovenia.

Mladenic, D., & Grobelnik, M. (1999). Feature selection for unbalanced class distribution and naive Bayes. *Machine Learning: Proceedings of the Sixteenth International Conference*, pp. 258–267.

Mooney, R. J., & Roy, L. (2000). Content-based book recommending using learning for text categorization. *Proceedings of the Fifth ACM Conference on Digital Libraries*, pp. 195–204.

Mosteller, F., & Wallace, D. L. (1964). *Inference and disputed authorship: The Federalist.* Reading, Massachusetts: Addison-Wesley.

Moulinier, I., Raskinis, G., & Ganascia, J.-G. (1996). Text categorization: a symbolic approach. *Fifth Annual Symposium on Document Analysis and Information Retrieval*, pp. 87–99.

Murray, G. D., & Titterington, D. M. (1978). Estimation problems with data from a mixture. *Applied Statistics*, *27*(3), 325–334.

Ng, A. Y. (1997). Preventing "overfitting" of cross-validation data. *Machine Learning: Proceedings of the Fourteenth International Conference*, pp. 245–253.

Nigam, K., & Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. *Ninth International Conference on Information and Knowledge Management*, pp. 86–93.

Nigam, K., Lafferty, J., & McCallum, A. (1999). Using maximum entropy for text classification. *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pp. 61–67.

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 792–799.

O'Neill, T. J. (1978). Normal discrimination with unclassified observations. *Journal of the American Statistical Association*, *73*(364), 821–826.

Pazzani, M. J., Muramatsu, J., & Billsus, D. (1996). Syskill & Webert: Identifying interesting Web sites. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 54–59.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1993). *Numerical recipes in C*. Cambridge University Press.

Ratsaby, J., & Venkatesh, S. S. (1995). Learning from a mixture of labeled and unlabeled examples with parametric side information. *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pp. 412–417.

Riloff, E. (1996). Automatically generating extraction patterns from untagged text. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 1044–1049.

Riloff, E., & Jones, R. (1999). Learning dictionaries for information extraction using multi-level boot-strapping. *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pp. 474–479.

Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, *11*(2), 416–431.

Robertson, S. E., & Sparck-Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, *27*(3), 129–146.

Rodriguez, M., Gomez-Hidalgo, J. M., & Agudo, B. D. (1997). Using WordNet to complement training information in text categorization. *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pp. 150–157.

Rose, K., Gurewitz, E., & Fox, G. C. (1990). Statistical mechanics and phase transitions in clustering. *Physical Review Letters*, *65*(8), 945–948.

Rose, K., Gurewitz, E., & Fox, G. C. (1992). Vector quantization by deterministic annealing. *IEEE Transactions on Information Theory*, *38*(4), 1249–1257.

Sahami, M. (1996). Learning limited dependence Bayesian classifiers. *KDD-96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 335–338.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A Bayesian approach to filtering junk e-mail. *Learning for Text Categorization: Papers from the AAAI Workshop*, pp. 55–62. Tech. rep. WS-98-05, AAAI Press.

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

Schohn, G., & Cohn, D. (2000). Less is more: Active learning with support vector machines. *Proceedings of the Seventeenth International Conference on Machine Learning*.

Schuurmans, D. (1997). A new metric-based approach to model selection. *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pp. 552–558.

Schuurmans, D., & Southey, F. (2000). An adaptive regularization criterion for supervised learning. *Proceedings of the Seventeenth International Conference on Machine Learning*.

Scott, S., & Matwin, S. (1998). Text classification using WordNet hypernyms. *Usage of WordNet in Natural Language Processing Systems: Proceedings of the Workshop*, pp. 45–52.

Sebastiani, F., Sperduti, A., & Valdambrini, N. (2000). An improved boosting algorithm and its application to text categorization. *Proceedings of the Ninth International Conference on Information and Knowledge Management*, pp. 78–85.

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee. *Machine Learning: Proceedings of the Fifth International Conference*, pp. 287–294.

Seymore, K., McCallum, A., & Rosenfeld, R. (1999). Learning hidden Markov model structure for information extraction. *Machine Learning for Information Extraction: Papers from the AAAI Workshop*. Tech. rep. WS-99-11, AAAI Press.

Shahshahani, B., & Landgrebe, D. (1994). The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, *32*(5), 1087–1095.

Shavlik, J., & Eliassi-Rad, T. (1998). Intelligent agents for web-based tasks: An advice-taking approach. *Learning for Text Categorization: Papers from the AAAI Workshop*, pp. 63–70. Tech. rep. WS-98-05, AAAI Press.

Stolcke, A., & Omohundro, S. M. (1994). *Best-first model merging for hidden Markov model induction*. Tech. rep. TR-94-003. ICSI, University of California, Berkeley.

Szummer, M., & Jaakkola, T. (2001). Kernel expansions with unlabeled data. *Advances in Neural Information Processing Systems 13*. To appear.

Titterington, D. M. (1976). Updating a diagnostic system using unconfirmed cases. *Applied Statistics*, *25*(3), 238–247.

Tong, S., & Koller, D. (2000). Support vector machine active learning with applications to text classification. *Proceedings of the Seventeenth International Conference on Machine Learning*.

Ueda, N., & Nakano, R. (1995). Deterministic annealing variant of the EM algorithm. *Advances in Neural Information Processing Systems 7*, pp. 545–552.

Vapnik, V. (1998). *Statistical learning theory*. New York: John Wiley and Sons.

Watanabe, S. (1969). *Knowing and guessing: A quantitative study of inference and information*. New York: John Wiley and Sons.

Wiener, E., Pedersen, J. O., & Weigend, A. S. (1995). A neural network approach to topic spotting. *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, pp. 317–332.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, *1*(1/2), 67–88.

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

Yang, Y., & Pedersen, J. O. (1997). Feature selection in statistical learning of text categorization. *Machine Learning: Proceedings of the Fourteenth International Conference*, pp. 412–420.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp. 189–196.

Zelikovitz, S., & Hirsh, H. (2000). Improving short-text classification using unlabeled background knowledge to assess document similarity. *Proceedings of the Seventeenth International Conference on Machine Learning*.

Zelterman, D. (1987). Goodness-of-fit tests for large sparse multinomial distributions. *Journal of the American Statistical Association*, *82*(398), 624–629.

Zhang, T., & Oles, F. J. (2000). A probability analysis on the value of unlabeled data for classification problems. *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 1191–1198.