Improving Text Classification by Shrinkage in a Hierarchy of Classes

Andrew McCallum[*†]
mccallum@justresearch.com

Ronald Rosenfeld[†]
roni@cs.cmu.edu

Tom Mitchell[†]
mitchell+@cs.cmu.edu

Andrew Y. Ng[‡]
ayn@ai.mit.edu

[*]Just Research
4616 Henry Street
Pittsburgh, PA 15213

[†]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[‡]MIT AI Lab
545 Technology Square
Cambridge, MA 02139

Abstract

When documents are organized in a large number of topic categories, the categories are often arranged in a hierarchy. The U.S. patent database and Yahoo are two examples.

This paper shows that the accuracy of a naive Bayes text classifier can be significantly improved by taking advantage of a hierarchy of classes. We adopt an established statistical technique called *shrinkage* that smoothes parameter estimates of a data-sparse child with its parent in order to obtain more robust parameter estimates. The approach is also employed in *deleted interpolation*, a technique for smoothing n-grams in language modeling for speech recognition.

Our method scales well to large data sets, with numerous categories in large hierarchies. Experimental results on three real-world data sets from UseNet, Yahoo, and corporate web pages show improved performance, with a reduction in error up to 29% over the traditional flat classifier.

1 Introduction

As the dramatic expansion of the World Wide Web continues, and the amount of on-line text grows, the development of methods for automatically categorizing this text becomes more important. A variety of recent work has demonstrated the success of statistical approaches for learning to classify text documents [Joachims 1997; Koller & Sahami 1997; Yang & Pederson 1997; Nigam *et al.* 1998]. These approaches, such as TFIDF [Salton 1991] and naive Bayes [Lewis & Ringuette 1994; McCallum & Nigam 1998], typically represent documents as vectors of words, and learn by gathering statistics from the observed frequencies of these words within documents belonging to the different classes. Because they rely on these learned word statistics, these approaches are data-intensive: they often require large numbers of hand-labeled training documents per class to achieve high classification accuracy.

This paper considers the question of how to scale up these statistical learning algorithms to tasks with a large number of classes and sparse training data per class. When humans organize extensive data sets into fine-grained categories, topic hierarchies are often employed to make the large collection of categories more manageable. Yahoo, the U.S. patent database, MEDLINE and the Dewey Decimal System are all examples of such hierarchies.

We present a technique that leverages these commonly available topic hierarchies in order to significantly improve classification accuracy, especially when the hierarchy is large and the training data for each class is sparse. We also present a method for exponentially reducing the amount of computation necessary for classification, while sacrificing only a small amount of accuracy.

Our approach applies a well-understood technique from Statistics called *shrinkage* that provides improved estimates of parameters that would otherwise be uncertain due to limited amounts of training data [Stein 1955; James & Stein 1961]. The technique exploits a hierarchy by "shrinking" parameter estimates in data-sparse children toward the estimates of the data-rich ancestors in ways that are provably optimal under the appropriate conditions. We employ a simple form of shrinkage that creates new parameter estimates in a child by a linear interpolation of all hierarchy nodes from the child to the root. The interpolation weights are learned by a form of Expectation Maximization [Dempster, Laird, & Rubin 1977]. This form of shrinkage is also applied in *deleted interpolation*, a technique for smoothing n-grams in language modeling for speech recognition [Jelinek & Mercer 1980].

Note that our approach to text classification in a hierarchy is quite different than work by Koller and Sahami [Koller & Sahami 1997]. Their *Pachinko Machine* employs the hierarchy by learning separate classifiers at each internal node of the tree, and then labeling a document by using these classifiers to greedily select sub-branches until it reaches a leaf. Their approach is shown to be helpful when documents are represented using a small subset (< 100 words) of the available vocabulary, and a different subset of the vocabulary is selected at each node of the hierarchy. However, their approach did not show improvement with larger vocabularies, and in many domains (including the domains studied in this paper) it has been established that large vocabulary sizes often perform best [Joachims 1997; Nigam *et al.* 1998; McCallum & Nigam 1998].

Somewhat surprisingly, it can be shown that a probabilistic form of Pachinko Machine, when trained using maximum likelihood estimates and a constant vocabulary, is equivalent to the simple non-hierarchical classifier [Mitchell 1998]. At each node in the hierarchy this non-deterministic version of the Pachinko Machine assigns each document probabilistically to all of its descendants, whereas the deterministic Pachinko Machine proposed by Koller and Sahami assigns each document to its single most probable descendant.

The remainder of this paper is structured as follows: we explain our probabilistic approach to text classification, and present the use of shrinkage in this context. Then we show experimental results on three real-world data sets, present related work, and close with a discussion of future work.

2 Probabilistic Framework

We approach the task of text classification in a Bayesian learning framework. We assume that the text data was generated by a parametric model, and use training data to calculate estimates of the model parameters. Then, equipped with these estimates, we classify new test documents by using Bayes rule to turn the generative model around and calculate the posterior probability that a class would have generated the test document in question. Classification then becomes a simple matter of selecting the most probable class given the document's words.

We assume that the data is generated by a mixture model, (parameterized by θ), with a one-to-one correspondence between mixture model components and (the observed) classes, $c_j \in \{\mathcal{C}\}$. This specifies that a document, d_i, is created by (1) selecting a class, c_j, according to the class priors, $\mathrm{P}(c_j|\theta)$, then (2) having the corresponding mixture component generate a document according to its own parameters, with distribution $\mathrm{P}(d_i|c_j;\theta)$. The marginal probability of generating document d_i is thus a sum of total probability over all mixture components:

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

A document is comprised of an ordered sequence of word events, drawn from a vocabulary V. We make the naive Bayes assumption: that the probability of each word event in a document is independent of the word's context given the class, and furthermore independent of its position in the document. Thus, each document d_i is drawn from a multinomial distribution with as many independent trials as the number of words in d_i. We also assume that document lengths, $|d_i|$, are independent of class. We write $w_{d_{ik}}$ for the word in position k of document d_i, where the subscript of w (in this case d_{ik}) indicates an index into the vocabulary. Then the probability of a document given its class is:

$$\mathrm{P}(d_i|c_j;\theta) = \mathrm{P}(|d_i|) \prod_{k=1}^{|d_i|} \mathrm{P}(w_{d_{ik}}|c_j;\theta). \quad (2)$$

Given the assumption about one-to-one correspondence between mixture model components and classes, the naive Bayes assumption, and the position independence assumption, the mixture model is composed of disjoint sets of parameters, θ_j, for each class c_j. This parameter set for each class, θ_j, is composed of probabilities for each word, w_t, such that $\theta_{jt} \equiv \mathrm{P}(w_t|c_j;\theta)$ and $\sum_{t=1}^{|V|} \theta_{jt} = 1$. The only other parameters in the model are the class prior probabilities, written $\theta_{0j} \equiv \mathrm{P}(c_j|\theta)$.

Given a set of labeled training documents, $\mathcal{D}$, we can calculate estimates for the parameters of the model that generated the documents. These estimates consist of straightforward counting of events, supplemented by standard Laplace 'smoothing' that primes each estimate with a count of one to avoid probabilities of zero. We define $N(w_t, d_i)$ to be the count of the number of times word w_t occurs in document d_i, and define $\mathrm{P}(c_j|d_i) \in \{0,1\}$ as given by the document's class label. Then, the estimate of the probability of word w_t in class c_j is

$$\hat{\theta}_{jt} \equiv \mathrm{P}(w_t|c_j;\hat{\theta}) = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} N(w_t,d_i)\mathrm{P}(c_j|d_i)}{|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|\mathcal{D}|} N(w_s,d_i)\mathrm{P}(c_j|d_i)}. \tag{3}$$

The class prior parameters are set by the maximum likelihood estimate:

$$\hat{\theta}_{0j} \equiv \mathrm{P}(c_j|\hat{\theta}) = \sum_{i=1}^{|\mathcal{D}|} \mathrm{P}(c_j|d_i)/|\mathcal{D}|. \tag{4}$$

Given estimates of these parameters calculated from the training documents, classification can be performed on test documents by calculating the posterior probability of each class given the words observed in the test document, and selecting the class with the highest probability. We formulate this by first applying Bayes rule, and then substituting for $\mathrm{P}(d_i|c_j;\theta)$ and $\mathrm{P}(d_i|\theta)$ using Equations 1 and 2.

$$\begin{aligned} \mathrm{P}(c_j|d_i;\hat{\theta}) &= \frac{\mathrm{P}(c_j|\hat{\theta})\mathrm{P}(d_i|c_j;\hat{\theta})}{\mathrm{P}(d_i|\hat{\theta})} \qquad (5) \\ &= \frac{\mathrm{P}(c_j|\hat{\theta})\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{ik}}|c_j;\hat{\theta})}{\sum_{r=1}^{|C|}\mathrm{P}(c_r|\hat{\theta})\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{ik}}|c_r;\hat{\theta})} \end{aligned}$$

Despite the fact that the mixture model and word independence assumptions are strongly violated with real-world data, naive Bayes performs text classification very well. Friedman and Domingos and Pazzani discuss why the violation of the word independence assumption sometimes does little damage to classification accuracy [Friedman 1997; Domingos & Pazzani 1997].

3 Hierarchical Classification

This section presents a method of improving our estimates of the model parameters by taking advantage of the hierarchy. We first briefly describe *shrinkage* in a general sense, then discuss its application to text classification in a hierarchy, and the mechanics of our algorithm.

Background on Shrinkage

We wish to estimate parameters $\theta_1, \ldots, \theta_{|\mathcal{C}|}$, (*i.e.* each class's probability distribution over words). The estimates $\hat{\theta}_j$ of θ_j can often be improved by shrinking

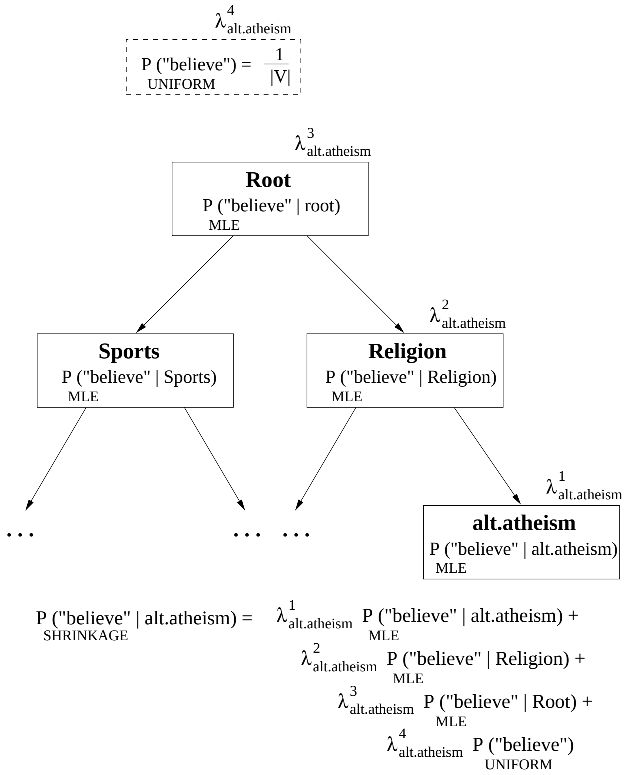

Figure 1: The new, shrinkage-based estimate of the probability of a word (*e.g.* "believe") given a UseNet class (*e.g.* alt.atheism) is a weighted sum of the maximum-likelihood estimates from the leaf to the root, and beyond the root to the uniform distribution over words.

each of them towards some common value. See Carlin and Louis [1996] for a recent summary of shrinkage. There are two justifications for shrinkage. First, if the quantities $\theta_1, \ldots, \theta_{|\mathcal{C}|}$ are thought to be similar, then they can regarded as draws from a common distribution. In this case, the shrinkage estimator is just the Bayes estimate. More surprisingly, even if the quantities are completely unrelated, and even if the data upon which each estimator is based are independent of each other, shrinkage estimators still reduce the risk of the estimators. This is a deep and counterintuitive fact discovered by Stein [1955] and James and Stein [1961].

Shrinkage for Text Classification

We use shrinkage to better estimate the probability θ_{jt} of word w_t given class c_j. For each node in our tree we construct a maximum likelihood (ML) estimate based on the data associated with that node (Equation 3 without the Laplace smoothing). An improved estimate for each leaf node is then derived by "shrinking" its ML estimate towards the ML estimates of all its ancestors, namely those estimates found along the path from that leaf to the root. Figure 1 illustrates this process. In statistical language modeling terms, we build a unigram model for each node in the tree, and smooth each leaf model by linearly interpolating it with all the models found along the path to the root.

The estimates along a path from the leaf to the root represent a tradeoff between specificity and reliability. The estimate at the leaf is the most specific (most pertinent, least biased), since it is based on data from that topic alone. However it is also the least reliable, since it is based on the smallest sample of data. The estimator at the root is the most reliable, but the least specific.

Since even the root contains a finite amount of data, it may estimate some rare words unreliably. We therefore extend the tree by adding, beyond the root, the uniform estimate. Thanks to the latter, we no longer need to smooth the individual ML estimates.

To ensure that the ML estimates along a given path are independent, we subtract each child's data from its parent's before calculating the parent's ML estimate. Thus the latter estimate is based on data that belongs to all the siblings of said child, but not to the child itself. Note that in this way, for any path from leaf to root, every datum in the tree is used in *exactly one* of the ML estimates, providing both independence among the estimates and efficient use of the training data.

Determining Mixture Weights

Given a set of ML estimates along the path from a leaf to the root (and beyond it, to the uniform estimate), how do we decide on the weights for interpolating (mixing) them? Let $\{\hat{\theta}_j^1, \hat{\theta}_j^2, \ldots, \hat{\theta}_j^k\}$ be k such estimates, where $\hat{\theta}_j^1 = \hat{\theta}_j$ is the estimate at the leaf, and $\hat{\theta}_j^k$ is the uniform estimate ($\hat{\theta}_{jt}^k = 1/|V|$ for all words w_t), and $k-2$ is the depth of class c_j in the tree. The interpolation weights among the ancestors of class c_j are written $\{\lambda_j^1, \lambda_j^2, \ldots, \lambda_j^k\}$, Where $\sum_{i=1}^k \lambda_j^i = 1$.

We write $\check{\theta}_j$ for the new estimate of the class-conditioned word probabilities based on shrinkage. The new estimate for the probability of word w_t given class c_j is

$$\check{\theta}_{jt} = \mathrm{P}(w_t|c_j;\check{\theta}_j) = \lambda_j^1\hat{\theta}_{jt}^1 + \lambda_j^2\hat{\theta}_{jt}^2 + \ldots + \lambda_j^k\hat{\theta}_{jt}^k. \quad (6)$$

We derive empirically optimal weights, λ_j^i, between the ancestors of c_j, by finding the weights that maximize the likelihood of some hitherto unseen "held-out" data. We use the fact that the likelihood of data according to the mixture model is a convex function of the weights (this falls out of Jensen's inequality), and thus attains a single, global maximum. We find that maximum for each leaf class, c_j, using the following iterative procedure:

Initialize: Set the λ_j's to some initial values, say $\lambda_j^i = \frac{1}{k}$ (any normalized non-zero initial values will do).

Iterate:

(1) Calculate the degree to which each estimate predicts the words w_t in the held-out set, $\mathcal{H}_j$, from class c_j:

$$\begin{aligned} \beta_j^i &= \sum_{w_t \in \mathcal{H}_j} \mathrm{P}(\hat{\theta}_j^i \text{ was used to generate } w_t) \\ &= \sum_{w_t \in \mathcal{H}_j} \frac{\lambda_j^i \hat{\theta}_{jt}^i}{\sum_m \lambda_j^m \hat{\theta}_{jt}^m} \end{aligned} \quad (7)$$

(2) Derive new (and guaranteed improved) weights by normalizing the β's:

$$\lambda_j^i = \frac{\beta_j^i}{\sum_m \beta_j^m} \quad (8)$$

Terminate: Upon convergence of the likelihood function (usually achieved within a dozen or so iterations).

This algorithm can be viewed as a particularly simple form of EM [Dempster, Laird, & Rubin 1977], where each datum is assumed to have been generated by first choosing one of the tree nodes in the path to the root, say $\hat{\theta}_j^i$ (with probability λ_j^i), then using that estimate to generate that datum. EM then maximizes the total likelihood when the choices of estimates made for the various data are unknown. The first step in the iterative part is thus the "E" step, and the second one is the "M" step.

While conceptually simple, this method makes inefficient use of the available training data by carving off some of it to be used as a held-out set. To overcome this problem, we modify the algorithm as follows: all the available data is used both to construct the ML estimates and to optimize the weights. However, as each document is used in the above algorithm, the ML estimates are modified to exclude its data, so as to make

# training documents	Class	Mixture Weights child	parent	g'parent	uniform
235	root/politics/talk.politics.guns	0.368	0.092	0.017	0.522
	root/politics/talk.politics.mideast	0.256	0.132	0.001	0.611
	root/politics/talk.politics.misc	0.197	0.213	0.026	0.564
	root/religion/alt.atheism	0.235	0.158	0.022	0.585
	root/religion/soc.religion.christian	0.181	0.189	0.052	0.578
	root/religion/talk.religion.misc	0.104	0.255	0.028	0.613
7497	root/politics/talk.politics.guns	0.801	0.089	0.048	0.061
	root/politics/talk.politics.mideast	0.859	0.061	0.010	0.071
	root/politics/talk.politics.misc	0.762	0.126	0.043	0.068
	root/religion/alt.atheism	0.766	0.174	0.043	0.018
	root/religion/soc.religion.christian	0.837	0.098	0.041	0.024
	root/religion/talk.religion.misc	0.663	0.226	0.049	0.062

Table 1: Mixture weights learned by EM for some nodes in the class hierarchy. Notice that when training data is sparse (top half of table), classes mix more strongly with their parents than when data is plentiful. Notice also that more 'generic' classes mix more strongly with their parents, *e.g.* talk.politics.misc's weight on its parent is higher than is talk.politics.guns's).

them independent of it. This method is very similar to the "leave-one-out" cross-validation commonly used in statistical estimation.

This technique of finding the optimal weights is routinely used in statistical language modeling to interpolate together different models (such as trigram, bigram, unigram and uniform), where it is known as "deleted interpolation" [Jelinek & Mercer 1980]. It was similarly used to interpolate estimates from nodes along a tree path in [Bahl *et al.* 1989]. This cross-validation approach to setting the mixture weights is not exactly the same style of shrinkage as Stein [1955] and James and Stein [1961], but is similar in spirit. In future work we will compare the different styles of shrinkage.

Table 1 shows a subset of the mixture weights learned by EM for a hierarchy based on UseNet articles.

4 Experimental Results

This section provides empirical evidence that shrinkage reduces text classification error by up to 29%. We also show that shrinkage helps most when training data is sparse and the number of classes is large. Finally, we demonstrate that dynamically pruning the tree can exponentially reduce computation time, at minimal loss of accuracy. Experiments are based on three different real-world data sets, one consisting of UseNet articles, and two of web pages.[1] The results are averages of ten cross-validation trials.

[1] All three data sets are available on-line. See http://www.cs.cmu.edu/~textlearning.

The Industry Sector hierarchy, made available by *Market Guide Inc.* (www.marketguide.com), consists of company web pages classified in a hierarchy of industry sectors. Using all classes at depth two results in 6440 web pages partitioned into 71 classes. In tokenizing the data we skip all MIME headers and HTML tags, use a stoplist, but do not stem. After removing tokens that occur only once, the corpus contains 1.2 million words, with a vocabulary of size 29964.

The Newsgroups data set, collected by Ken Lang, contains about 20,000 articles evenly divided among 20 UseNet discussion groups [Joachims 1997]. Several of the topic classes are quite confusable: five of them are about computers; three discuss religion. From this data set, we build a two-level hierarchy from the 15 classes that fit into the following top level categories: vehicles, computers, politics, religion and sports. We tokenize the data in the same way as above. The resulting data set, after removing words that occur only once, contains 1.7 million words, and a vocabulary size of 52309.

We gathered the entirety of the Yahoo 'Science' hierarchy in July 1997. The web pages pointed to by Yahoo are divided into 264 disjoint classes containing 14831 pages as result of descending to deeper nodes of Yahoo's hierarchy until each class contains less than 200 documents, and then removing classes with fewer than 20 documents. After tokenizing as above and removing stopwords and words that occur only once, the corpus contains 3.0 million words, with a vocabulary size of 76624.

Feature selection, when used, is performed by select-

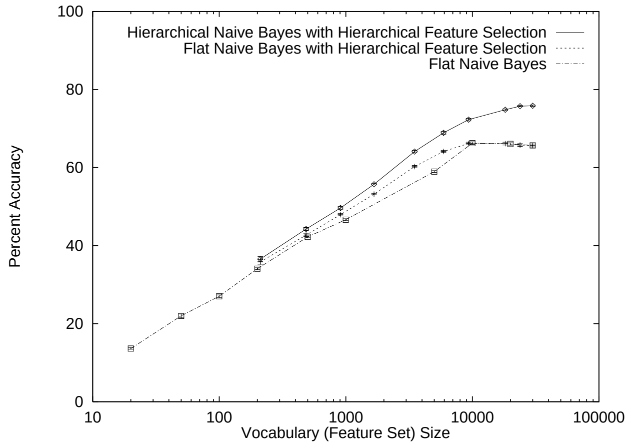

Figure 2: Classification accuracy on the Industry Sector data set with varying vocabulary size in the horizontal axis. The tiny vertical bars at each data point indicate standard error. Performance is best with the full vocabulary, where shrinkage reduces error by almost one-third.

ing the words that have highest mutual information with the class. A previous study found this method to be the best for text among several common methods [Yang & Pederson 1997]. In addition to selecting features by the traditional, flat use of mutual information, we also use the hierarchy for feature selection. *Hierarchical feature selection* selects equal numbers of top words by mutual information *at each internal node of the tree*, using the node's immediate children as the classes. This corresponds to Koller and Sahami's hierarchical feature selection with zero dependencies [Koller & Sahami 1997], except that we define the total vocabulary to be the union of all the vocabularies chosen by the internal nodes. The union is necessary so that the models we will mix share the same event space.

Hierarchical classification improves accuracy

Figure 2 shows classification accuracy on the Industry Sector data set with 50-50 train-test splits while varying vocabulary size. No partial credit is given for classification into neighbors of the true class.

First, note that larger vocabulary sizes generally perform better; this is consistent with previous results of naive Bayes on several other data sets [Joachims 1997; Nigam *et al.* 1998; McCallum & Nigam 1998]. Second, note that Hierarchical Feature Selection somewhat improves the performance of flat naive Bayes in the mid-range of feature selection—at about 5000 words, traditional, flat feature selection obtains 59% accuracy, while hierarchical feature selection reaches 64%. Third, and most importantly, observe that shrinkage improves classification accuracy across the board, making the largest improvement at the full, unpruned vocabulary size, where it achieves 76% accuracy. In comparison, the flat classifier reaches its best performance of 66% at about 10000 words. This difference represents a 29% reduction in classification error. We maintain that low-frequency words contribute significantly to correct classifications, and that shrinkage helps reduce variance of the estimates in the larger parameter space that results from the larger vocabulary.[2]

Shrinkage helps more when training data is sparse.

Figure 3 shows accuracy on the Newsgroups data set with the full vocabulary, while varying amount of training data. Our experiments indicate that accuracy in this domain is highest with no feature selection, (*i.e.* using the full vocabulary), for both flat and hierarchical classifiers, even with small amounts of training data.

It is interesting to see that hierarchical modeling provides less improvement on this data set than it does in the Industry Sector corpus. We expect that this is due to the significantly reduced branch-out factor in this smaller hierarchy. Unlike the Industry Sector hierarchy, in which the mean number of siblings is six, here the mean number of siblings is three. Thus each child has fewer siblings and less data from which to "borrow strength."

The second expected result, exhibited in Figure 3, is that shrinkage provides more improvement when the amount of training data is small, and that shrinkage reduces variance in the classifications; (notice larger error bars on the 'flat classification' curve). If each class had an infinite amount of training data, accurate parameter estimates could be obtained for each class independently; however, when training data is sparse, estimates are improved by using shrinkage to smooth a class's parameters with its ancestors.

The two findings that (1) shrinkage allows the use of helpful large vocabulary sizes, and (2) shrinkage improves performance more when training data is sparse, are both confirmed by our experiments with the Yahoo data set. Figure 4 shows classification accuracy

[2]Large vocabularies need not be a computational concern. In our experiments, with the largest vocabulary, it takes only 216 seconds to classify 3220 Industry Sector documents and write the results to disk. In comparison, the smallest vocabulary takes 208 seconds—a difference of 0.002 seconds per document on average.

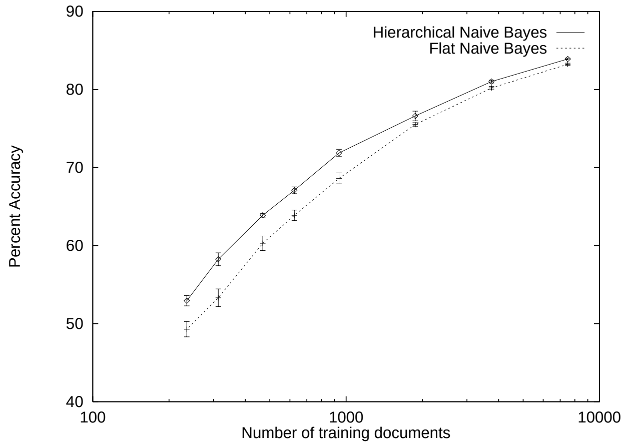

Figure 3: Classification accuracy on the Newsgroups data set with varying amounts of training data. The vertical axis is zoomed for magnification of the error bars. Overall, hierarchical modeling provides less improvement than it does in the Industry Sector data set because the hierarchy is much smaller. Notice, however, that, as expected, shrinkage helps more when there is less training data.

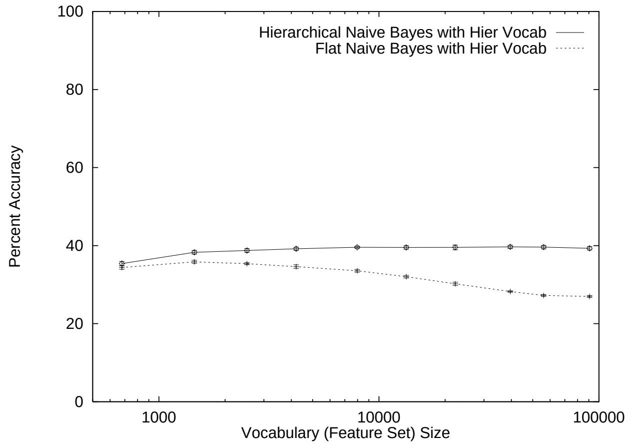

Figure 4: Classification accuracy on the Yahoo Science data set with varying vocabulary sizes. Tiny vertical bars at each data point indicate standard error. The large number of classes and noisy data make this task difficult.

on the Science hierarchy as a function of vocabulary size, again, with no partial credit for near misses. Flat naive Bayes reaches its highest accuracy of 35.8% at a relatively small vocabulary size of 1449. Hierarchical classification always performs better than flat, but attains its best accuracy of 39.7% at a much larger vocabulary size of 39553. The improvement in accuracy is not as dramatic here as with the Industry Sector data set, perhaps because the Yahoo set is more noisy (being gathered automatically rather than by hand, and containing many documents that are simply timeout messages or pointers to moved pages), and because Yahoo has many classes with overlapping or closely neighboring definitions.[3] However, it is interesting to note that among those classes with small quantities of training data, shrinkage improves performance more significantly. Among those 151 classes with 50 documents or less, shrinkage improves accuracy by 10%, from 27% to 37%. Among those 50 classes containing more than 100 documents, shrinkage does not improve accuracy, both obtaining 45%.

This result indicates that shrinkage would be all the more important if we attempted to classify documents into Yahoo's deepest leaf categories instead of into the somewhat coalesced and pruned version used here and defined at the beginning of this section. However, this would result in thousands of classes—quite a computational burden. Next we describe how the hierarchy itself can be used to ease this burden.

[3] Using more complex Bayesian classifiers that capture more dependancies than naive Bayes may help this last problem. The larger number of paramters in these models will make training data even more sparse, and this suggests that the use of shrinkage would be all the more important.

Pruning the tree for increased computational efficiency

In addition to improving accuracy, the class hierarchy can also be leveraged to improve computational efficiency. The classifier can avoid calculating $\mathrm{P}(c_j|d_i)$ for a majority of the classes (leaves of the tree) by pruning the tree dynamically during the classification of each document. Like the *Pachinko Machine* [Koller & Sahami 1997] we can classify the document at internal nodes of the tree, and choose only to calculate probabilities for classes underneath the branches selected by these higher-level, coarse-grained classifiers. Note, however, that when we do this, each "pruning classification" at the interior of the tree is an opportunity for error, and the deeper the hierarchy the more the opportunities for error will compound.

As expected, our experimental results show that performing this pruning does indeed reduce classification accuracy. However, one may be willing to accept this reduction in exchange for the exponential reduction in the amount of computation necessary for classification. On the Industry Sector data set, averaged over ten runs, pruning that removes from consideration all but a single branch at each interior node reaches 70.0% accuracy, more than 5% points lower than without pruning. However, unlike the *Pachinko Machine*, our paradigm allows for the comparison of classification scores from leaves that do not share the same parent. Thus we can also prune less aggressively. Pruning that keeps two branches attains 74.3%. And pruning to three branches achieves 75.2%. This last result is only half a percent less than the 75.8% obtained by the full evaluation of the tree without pruning. The same approach could also be used for Yahoo.

5 Related Work

Shrinkage estimation is now considered standard methodology in Statistics. It is used routinely in a vast array of problems and its theoretical properties have been studied from both the Bayesian and frequentist points of view. A good discussion with ample references and examples is contained in [Carlin & Louis 1996]. Although MacKay and Peto [1994] do not use the term "shrinkage" in their paper, they use this Bayesian style of shrinkage in their hierarchical Dirichlet model for n-grams.

Shrinkage in the cross-validation style was first used to derive a language model in [Jelinek & Mercer 1980], where it is known as *deleted interpolation*. Interpolation of language models along the path of a tree is described in [Bahl *et al.* 1989]. More recently, Seymore and Rosenfeld [1997] classified a speech recognizer's output into multiple topics, then used an automatically derived "topic tree" to interpolate the models associated with appropriate nodes up that tree.

A variety of work in the Information Retrieval and Machine Learning communities has demonstrated the success of statistical approaches for learning to classify text documents. Naive Bayes has been used for text classification, and due to its probabilistic foundations, been applied in several extensions [Lewis & Ringuette 1994; Joachims 1997; Nigam *et al.* 1998].

An earlier approach to hierarchical document classification, the *Pachinko Machine*, has been proposed by Koller and Sahami [1997]. Their method differs significantly from shrinkage. The Pachinko Machine classifies documents at internal nodes of the tree, and greedily selects sub-branches until it reaches a leaf. Since classification errors at internal nodes compound, the accuracy at all the internal nodes must be very high in order for overall accuracy to be higher than a flat classifier (especially for deeper hierarchies). We experimented with schemes that allow a lower node to "reject" a document and send it back up the tree for re-classification, but did not find these to work well. Koller and Sahami present results with small vocabularies (less than 100 words); however, other results in the literature indicate that large vocabulary sizes often have higher accuracy [Joachims 1997; Nigam *et al.* 1998]. A possible explanation for the discrepancy is that Koller and Sahami use a multivariate Bernoulli model while we use a multinomial model [Sahami, Personal Communication]. In our experiments we have found multinomials to outperform Bernoullis [McCallum & Nigam 1998]. Our use of shrinkage has allowed us to more robustly keep large vocabulary sizes, which we believe are necessary for classifying large data sets with large numbers of diverse classes.

Another learning method that uses EM to set mixture weights among ancestors in a hierarchy is *Adaptive Mixtures of Probabilistic Transducers* [Singer 1997]. Each node in a hierarchy that represents a history-window is linearly mixed with its parent, which in turn, is mixed with its parent. The model is applied with success to noun phrase recognition.

Hofmann and Puzicha's [1998] Hierarchical Asymetric Clustering Model (HACM) performs unsupervised clustering with a mixture model in which EM is also used to set weights among the ancestors in a hierarchy.

6 Conclusions

This paper has examined the use of class hierarchies for improving text classification. As the amount of on-line text increases and the number of topic categories into which it is organized grows, hierarchies are becoming an increasingly prevalent way to make a collection of categories manageable. Thus, the need for good text classification algorithms that take advantage of these hierarchies becomes more important.

In this paper we demonstrate that shrinkage with a class hierarchy improves parameter estimation, and can reduce text classification error by up to 29%. Because shrinkage helps especially when there is sparse training data, shrinkage should be all the more beneficial as we scale up to larger, higher-resolution, deeper hierarchies with more classes that require larger vocabularies.

We also show that a class hierarchy can be used to exponentially reduce the amount of computation required to classify documents, and that we can do so without sacrificing significant classification accuracy.

In future work, we will investigate the use of shrinkage to learn more complex Bayesian models with less restrictive assumptions than naive Bayes. The improvements due to shrinkage should be increasingly strong as we move to models that have more parameters, and thus sparser training data. We will also explore alternative methods of shrinkage, including the Bayesian methods in the style of James and Stein. We plan to work with a related approach that uses EM to cluster the data in a parent, and then allows the child to mix with the different clusters independently. In other ongoing work we are studying the advantages of using EM not only to set the mixture weights, but also redistribute individual words of training data among the nodes on the path from the leaf to the root.

Lastly, we plan to explore ways to learn the class hierarchy—investigating methods that specifically aim to increase classification accuracy. In early experiments, it appears that when the learner is not explicitly given a hierarchy, then even using the "trivial" hierarchy (each class being a leaf off the root) does better than the flat classifier, though not as well as when we are given a "non-trivial" hierarchy. Furthermore, using a "bad" or scrambled hierarchy also does better than the flat classifier—the mixture weights are set by EM to mimic the trivial hierarchy.

Acknowledgments

Larry Wasserman contributed many valuable discussions, pointers to the statistics literature, and comments for this paper. Kamal Nigam provided numerous helpful suggestions on earlier drafts. We thank *Market Guide, Inc.* for permission to use their Industry Sector hierarchy, and Mark Craven for gathering its data from the web. We thank *Yahoo!* for permission to use their data. This research was supported in part by the Darpa HPKB program under contract F30602-97-1-0215. A. Y. Ng is supported by the NSF under contract number ASC-92-17041.

References

Bahl, L.; Brown, P.; deSouza, P.; and Mercer, R. 1989. A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing* 37:1001–1008.

Carlin, B., and Louis, T. 1996. *Bayes and Empirical Bayes Methods for Data Analysis.* Chapman and Hall.

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

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

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

Hofmann, T., and Puzicha, J. 1998. Statistical models for co-occurrence data. Technical report, Artificial Intelligence Laboratory and Center for Biological and Computational Learning, MIT. AI Memo 1625, CBCL Memo 159.

James, W., and Stein, C. 1961. Estimation with quadratic loss. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability 1*, 361–379. University of California Press.

Jelinek, F., and Mercer, R. 1980. Interpolated estimation of Markov source parameters from sparse data. In Gelsema, S., and Kanal, L. N., eds., *Pattern Recognition in Practice*, 381–402.

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

Koller, D., and Sahami, M. 1997. Hierarchically classifying documents using very few words. In *ICML-97: Proceedings of the Fourteenth International Conference on Machine Learning*, 170–178. Morgan Kaufmann.

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

MacKay, D., and Peto, L. B. 1994. A hierarchical dirichlet language model. *Natural Language Engineering* 1(1).

McCallum, A., and Nigam, K. 1998. A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization.* http://www.cs.cmu.edu/~mccallum.

Mitchell, T. M. 1998. Conditions for the equivalence of hierarchical and flat Bayesian classifiers. http://www.cs.cmu.edu/~tom/hierproof.ps.

Nigam, K.; McCallum, A.; Thrun, S.; and Mitchell, T. 1998. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI-98.*

Salton, G. 1991. Developments in automatic text retrieval. *Science* 253:974–979.

Seymore, K., and Rosenfeld, R. 1997. Using story topics for language model adaptation. In *Eurospeech.*

Singer, Y. 1997. Adaptive mixtures of probabilistic transducers. *Neural Computation* 9(8).

Stein, C. 1955. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability 1*, 197–206. University of California Press.

Yang, Y., and Pederson, J. 1997. Feature selection in statistical learning of text categorization. In *ICML-97*, 412–420.