Submitted (3/98) to the *15th International Conference on Machine Learning (ICML-98).*

Improving Text Classification by Shrinkage in a Hierarchy of Classes

Andrew McCallum[*†] **Ronald Rosenfeld**[†] **Tom Mitchell**[†] **Andrew Ng**[‡]
mccallum@justresearch.com *roni@cs.cmu.edu* *mitchell+@cs.cmu.edu* *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 Tech 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.

Keywords:
Text Classification
Information Retrieval
Hierarchical Modeling
Statistical Language Modeling
Bayesian Learning

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 and Sahami, 1997; Yang and Pederson, 1997; Nigam *et al.*, 1998]. These approaches, such as TFIDF [Salton, 1991] and naive Bayes [Lewis and Ringuette, 1994], 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 and 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 *et al.*, 1977]. This form of shrinkage is also applied in *deleted interpolation*, a technique for smoothing n-grams in language modeling for speech recognition [Jelinek and Mercer, 1980].

Note that our approach to text classification in a hierarchy is quite different than work by Koller and Sahami [Koller and 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 it has been established that large vocabulary sizes perform best [Joachims, 1997; Nigam *et al.*, 1998]. Somewhat surprisingly, it can be shown that a pure form of Pachinko Machine classification using maximum likelihood estimates and a constant vocabulary is in fact equivalent to performing non-hierarchical classification [Mitchell, 1998].

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 and Naive Bayes

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, armed 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.

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 classes, $c_j \in \{\mathcal{C}\}$. This dictates 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:

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

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 also 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 write $w_{d_{ik}}$ for the word in position k of document d_i, where the subscript of w indicates an index into the vocabulary. Then, the probability of a document given its class is:

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

Given the assumption about one-to-one correspondence between mixture model components and classes, and the naive Bayes assumption, and 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, such that $\theta_{jt} = \mathrm{P}(w_t|c_j;\theta)$ and $\sum_t \theta_{jt} = 1$. The only other parameters in the model are the class prior probabilities, written $\theta_{0j} = \mathrm{P}(c_j|\theta)$.

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 $\mathrm{P}(c_j|d_i) = \{0, 1\}$ as given by the document's class label. Then, the estimate of the probability of word w_t in class c_j is

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

The class prior parameters are set by the maximum likelihood estimate, $\hat{\theta}_{0j} = \mathrm{P}(d_i|c_j;\theta) = \sum_{i=1}^{|\mathcal{D}|} \mathrm{P}(c_j|d_i)/|\mathcal{D}|$.

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 evidence of 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.

$$\begin{aligned} \mathrm{P}(c_j|d_i;\theta) &= \frac{\mathrm{P}(c_j|\theta)\mathrm{P}(d_i|c_j;\theta)}{\mathrm{P}(d_i|\theta)} \\ &= \frac{\mathrm{P}(c_j|\theta)\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{ik}}|c_j;\theta)}{\sum_{r=1}^{|C|}\mathrm{P}(c_r|\theta)\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{ik}}|c_r;\theta)} \end{aligned} \tag{4}$$

Despite the fact that the mixture model and word independence assumptions are strongly violated with real-world data, naive Bayes performs classification very well. Domingos and Pazzani discuss why the violation of the word independence assumption does little damage to classification accuracy [Domingos and 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), and for each parameter θ_j we have some estimate $\hat{\theta}_j$. The estimators can often be improved by shrinking 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 of a word given a class, $\hat{\theta}_{jt}$. 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. 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 usually based on the least amount 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 $\hat{\theta}_j^0 = \hat{\theta}_j$ is the estimate at the leaf, and $k-1$ is the depth of class c_j in the tree. The interpolation weights among the ancestors of class c_j are written $\{\lambda_j^0, \lambda_j^1, \ldots, \lambda_j^k\}$, Where $\sum_i \lambda_j^i = 1$.

We write $\tilde{\theta}_j$ for the new estimate of the class-conditioned word probabilities

based on shrinkage. It is:

$$\tilde{\theta}_j = \mathrm{P}(W|c_j;\tilde{\theta}_j) = \lambda_j^0\hat{\theta}_j^0 + \lambda_j^1\hat{\theta}_j^1 + \ldots + \lambda_j^{k-1}\hat{\theta}_j^{k-1} + \lambda_j^k\frac{1}{|V|}. \tag{5}$$

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 initial values will do).

Iterate:

(1) Calculate, over all words w_t in the held-out set:

$$\beta_j^i = \sum_l \mathrm{P}(\hat{\theta}_j^i \text{ was used to generate } w_t) = \sum_l \frac{\lambda_j^i\hat{\theta}_j^i(w_t)}{\sum_m \lambda_j^m\hat{\theta}_j^m(w_t)} \tag{6}$$

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

$$\lambda_j^i = \frac{\beta_j^i}{\sum_m \beta_j^m} \tag{7}$$

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 *et al.*, 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 them independent of it. This method, known as "leave-one-out" or "jacknifing", is 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 and Mercer, 1980].

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 per class data is sparse and the number of classes is large. Finally, we demonstrate that dynamically pruning the tree exponentially reduces 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.

The Industry Sector hierarchy, made available by *Market Guide Inc.* (www.marketguide.com) with data collected by Mark Craven, consists of company web pages classified in a hierarchy of industry sectors. Because our implementation is currently hardcoded to handle hierarchies of depth two only, we remove classes that have depth different from two. The result is 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.

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

We gathered the entirety of the Yahoo 'Science' hierarchy in July 1997. The web pages are divided into 95 disjoint classes containing 13589 pages as result of coalescing classes of hierarchy depth greater than two, and removing those classes with fewer than 40 documents. After tokenizing as above and removing stopwords and words that occur only once, the corpus contains 2.5 million words, with a vocabulary size of 44383. We also gathered the Yahoo 'Health' hierarchy, resulting in 80 classes, 3723 documents and 0.6 million words.

Feature selection, when used, is performed by selecting 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 and 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 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 and Sahami, 1997], except that we define the total vocabulary to be the union of all the vocabularies chosen by the internal nodes.

Hierarchical classification improves accuracy.

Figure 1 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]. 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]

[2]Large vocabularies need not be a computational concern. In our experiments, with

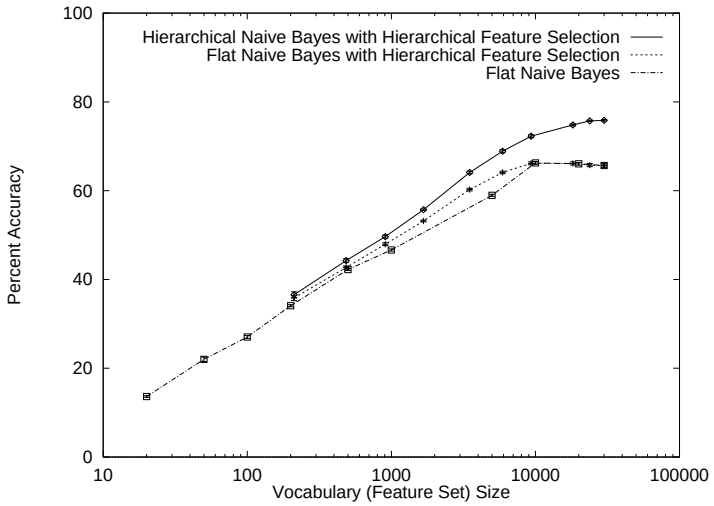

Figure 1: Classification accuracy on the Industry Sector data set with varying vocabulary size in the horizontal axis. The tiny vertical bars at each data point indicate standard error. Performance is best with the full vocabulary, where shrinkage reduces error by almost one-third.

Shrinkage helps more when training data is sparse.

Figure 2 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, 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 less data with which to "borrow strength."

The second expected result, confirmed by the graph in Figure 2, 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

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.

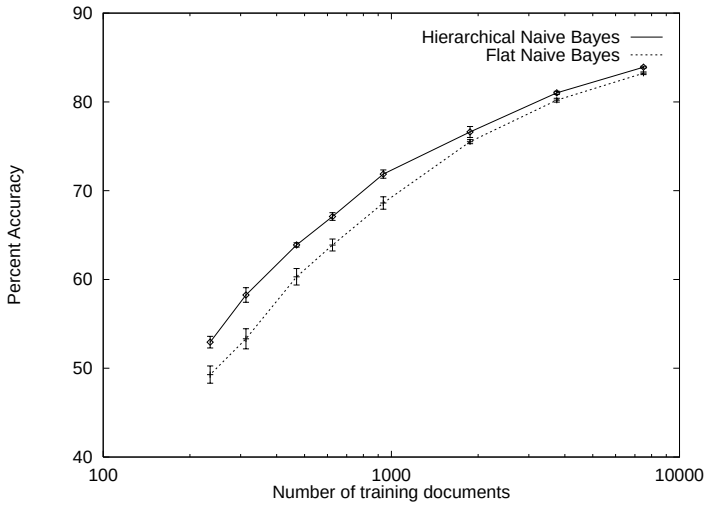

Figure 2: Classification accuracy on the Newsgroups data set with varying amounts of training data. The vertical axis is zoomed for magnification of the error bars. Overall, hierarchical modeling provides less improvement than it does in the Industry Sector data set because the hierarchy is much smaller. Notice, however, that, as expected, shrinkage helps more when there is less training data.

independently; however, when training data is sparse, estimates are improved by using shrinkage to smooth a class's parameters with its ancestors.

This finding is also confirmed by our experiments with the Yahoo data set. Figure 3(a) shows classification accuracy on the Science hierarchy as a function of vocabulary size. Best performance for hierarchical and flat classification is about equal. However, among the 51 classes with less than 50 training documents, shrinkage provides a 6% improvement in accuracy—from 39% for flat, to 45% for hierarchical (each at their best-performing vocabulary size). Among those classes with a large amount of training data, shrinkage either has no effect or hurts slightly. (This is probably due to the fact that that shrinkage was performed using linear interpolation, which is only optimal under certain conditions that are not satisfied here; we believe the performance of shrinkage would be improved on these data-heavy classes by using more complex standard Bayes or Empirical Bayes techniques as discussed in [Carlin and Louis, 1996], and plan to investigate this in future work.) Because the majority of the training and testing documents fall into a few classes, accuracy averaged over all testing documents is not improved by shrinkage.

Some users of document classification may care about accuracy in the small

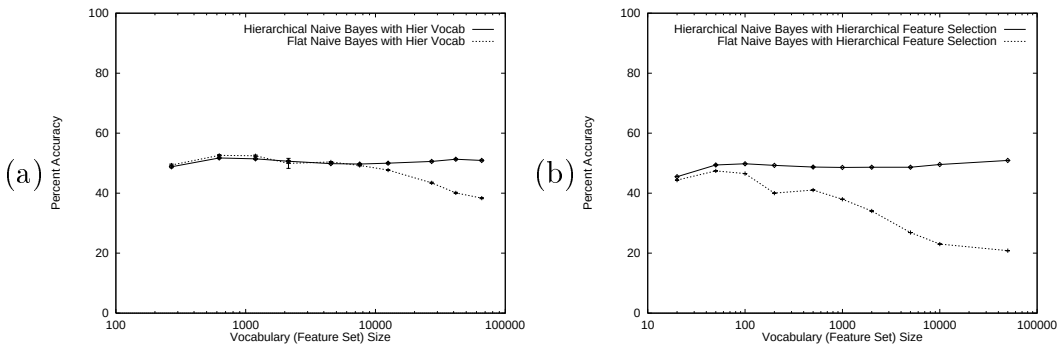

Figure 3: Classification accuracy on the Yahoo data set with varying vocabulary sizes. **(a)** Yahoo Science. **(b)** Yahoo Science with accuracy averaged over classes instead of documents.

classes as much as accuracy in the large classes. Figure 3(b) plots accuracy averaged over classes instead of documents. Here we see that, even with the presence of large classes, shrinkage increases accuracy by 3.5%.

Note that, when our shrinkage implementation can handle hierarchies of depth greater than two, we will be able to represent Yahoo's deepest leaf classes. Then all of the classes will have a small amount of training data because it is the nature of Yahoo (and often other hierarchy creators) to split categories when they become large.

We simulate a shallow and thin version of this effect by testing shrinkage on the Yahoo Health hierarchy, coalesced at depth two, but with the 13 (out of 93) classes that have more than 100 documents removed. As measured with traditional accuracy averaged over test documents, shrinkage increases classification accuracy by 5% points. In future experiments with the full, deeper and wider tree, we expect this effect to increase.

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 and 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.

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 and Louis, 1996].

Shrinkage and leave-one-out techniques were first used to derive a language model in [Jelinek and Mercer, 1980], where they were 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 and 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 [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 binomial model while we use a multinomial model [Sahami, Personal Communication]. In our experiments we have found multinomials to outperform binomials. 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.

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. The improvements due to shrinkage should also be increasingly strong as we move away from simple naive Bayes classifiers and on to more complex probabilistic models that have more parameters, and thus sparser training data.

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 explore alternative methods of shrinkage, including improved methods for setting mixture weights. We will also use EM to cluster the data in a parent, and allow the child to mix with the EM clusters independently. In addition, we plan to explore ways to learn the class hierarchy—investigating methods that specifically aim to increase classification accuracy.

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.

References

[Bahl *et al.*, 1989] Lalit Bahl, Peter Brown, Peter deSouza, and Robert Mercer. A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37:1001–1008, 1989.

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

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

[Domingos and Pazzani, 1997] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. *Machine Learning*, 29:103–130, 1997.

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

[Jelinek and Mercer, 1980] Fred Jelinek and Robert Mercer. Interpolated estimation of markov source parameters from sparse data. In S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice*, pages 381–402, 1980.

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

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

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

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

[Nigam *et al.*, 1998] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. Learning to classify text from labeled and unlabeled documents. In *Submitted to AAAI-98*, 1998. http://www.cs.cmu.edu/∼mccallum.

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

[Seymore and Rosenfeld, 1997] Kristie Seymore and Ronald Rosenfeld. Using story topics for language model adaptation. In *Eurospeech*, September 1997.

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

[Stein, 1955] C. Stein. 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*, pages 197–206. University of California Press, 1955.

[Yang and Pederson, 1997] Yiming Yang and Jan Pederson. Feature selection in statistical learning of text categorization. In *ICML-97*, pages 412–420, 1997.

[Yang, 1997] Yiming Yang. An evaluation of statistical approaches to text categorization. Technical Report CMU-CS-97-127, Computer Science Department, Carnegie Mellon University, 1997.