

Text Classification in a Hierarchical Mixture Model for Small Training Sets

Kristina Toutanova

Xerox PARC and
Stanford University
Stanford, CA 94305
kristina@cs.stanford.edu

Francine Chen

Xerox PARC
3333 Coyote Hill Rd
Palo Alto, CA 94304
fchen@parc.xerox.com

Kris Papat

Xerox PARC
3333 Coyote Hill Rd
Palo Alto, CA 94304
popat@parc.xerox.com

Thomas Hofmann

Dept. of Computer Science
Brown University
Providence, RI 02912
th@cs.brown.edu

ABSTRACT

Documents are commonly categorized into hierarchies of topics, such as the ones maintained by Yahoo! and the Open Directory project, in order to facilitate browsing and other interactive forms of information retrieval. In addition, topic hierarchies can be utilized to overcome the sparseness problem in text categorization with a large number of categories, which is the main focus of this paper. This paper presents a *hierarchical mixture model* which extends the standard naive Bayes classifier and previous hierarchical approaches. Improved estimates of the term distributions are made by differentiation of words in the hierarchy according to their level of generality/specificity. Experiments on the Newsgroups and the Reuters-21578 dataset indicate improved performance of the proposed classifier in comparison to other state-of-the-art methods on datasets with a small number of positive examples.

1. INTRODUCTION

The number of online documents is vastly growing in size, making the ability to automatically organize and categorize documents increasingly important. In text categorization or text classification, one is concerned with annotating a document with a single or several classes that describe the contents of the document. Here, text categories are usually defined by the topics discussed in each document. Although there are repositories for which large collections of documents have been manually labeled with category information (e.g., Yahoo!, Open Directory, MeSH, and U.S. Patents) it is desirable to automatically update these hierarchies and to be able to add new categories for which only very few human labeled exemplars are given.

There are a number of statistical methods specifically developed to handle sparse data, including shrinkage [14] and deleted interpolation [7]. These methods make use of term distributions estimated for more general, coarser text classes to provide better, smoothed estimates of class conditional term distributions, $P(w|c)$ where w is a term and c is a class. The more general classes are typically obtained through the use of a given taxonomy or topic hierarchy. Recent work on text classification has used topic hierarchies to reduce variance in parameter estimation [14, 17], for successive refinement of classification decisions [9, 2], and as an integral part of a classifier [1].

In this paper we propose a novel approach to using a topic hierarchy for text classification which is most closely related to the method for shrinkage in a hierarchy of classes presented in [14]. From here on, we will refer to this method of shrinkage using a hierarchy of classes [14] as Hierarchical Shrinkage Model for brevity. In [14], the class-conditional word probabilities, $P(w_t|c_j)$ for word w_t and class c_j , are estimated as:

$$P(w_t|c_j; \theta) = \lambda_j^1 \theta_{jt}^1 + \lambda_j^2 \theta_{jt}^2 + \dots + \lambda_j^k \theta_{jt}^k$$

where $\theta_{jt}^i = P(w_t|c_j)$ is the maximum likelihood estimate based on the data that belongs to all classes that are successors of the node i , but not the actual class c_j itself. Our Bayesian approach uses a predefined hierarchy of topics, but is able to automatically differentiate terms according to their specificity/generality, by reformulating the cluster-abstraction model [4] in a supervised setting. Compared to [14] our model achieves selective shrinkage over terms. This performs smoothing while maintaining a higher degree of discriminability between the classes. The differences between our model and [14] and [4] are discussed in more detail in section 2.

Following the paradigm of naive Bayesian classification, a document's empirical term distribution, that is, the vector of term frequencies representing a document, is matched against a collection of class-conditional distributions, $P(w|c)$, which have been estimated from training data. The accuracy of classification depends on how the matching is performed, but also in large part on the accuracy of the esti-

mates for the class-conditional term probabilities, $P(w|c)$. Since the vocabulary size can be large, the accuracy of these estimates is often severely limited by the sparsity of the available training data. This paper considers combating this sparseness by *hierarchical shrinkage*; more specifically, by convexly combining conditional term distributions in a hierarchical mixture model.

This paper also presents experiments which compare the performance of our method to the performance of several related statistical approaches - Naive Bayes, the Hierarchical Shrinkage Model [14], Probabilistic Latent Semantic Analysis [5], as well as two other well-established models with good performance for text classification - k -NN and SVMs. We report results on two widely used benchmarks for text classification - a subset of the 20 Newsgroups dataset in which each document has a single correct class, and Reuters-21578, which requires multi-label classification. Our experiments show that besides desirable theoretical properties, the proposed model has clear advantages in classification accuracy over the non-hierarchical models and the hierarchical shrinkage model, in particular in cases where only a small number of training documents per class is available.

2. HIERARCHICAL MIXTURE MODEL

In this section, we compare our model to that of [14] and [4] and then describe the assumptions and the technical details of the parameter estimation procedure of the Hierarchical Mixture Model which is the focus of this paper.

2.1 Comparison to the Hierarchical Shrinkage and Cluster Abstraction Models

Although our model has similarities to the Hierarchical Shrinkage Model [14] and the Cluster Abstraction Model [4], there are significant differences. As in the Hierarchical Shrinkage model, we assume that a predefined hierarchy of text categories is given. Based on this hierarchy, we will define a generative probability model for documents. The model parameters are learned through an Expectation-Maximization procedure which aims at maximizing the model likelihood on training data. Once the model parameters have been estimated, new documents are classified following Bayes' rule by computing (and maximizing) posterior probabilities over categories, $P(c|d)$, given the words of the document as observations.

The major difference between our model and [14] is that the hierarchy of topics is not only used to provide better estimates for class-conditional term probabilities, $P(w|c)$ for word w and class c , for data-sparse classes, but also to obtain a differentiation of words in the hierarchy according to their level of generality/specificity. In the hierarchical shrinkage model, interior nodes in the hierarchy represent coarser views of the data which are obtained by a simple pooling scheme of term counts. In the hierarchical mixture model, the inner nodes take a more accentuated role

and represent abstraction levels with their corresponding specific vocabulary. It is assumed that each word in a document is generated from some node (abstraction level) on the path from the document class' (terminal) node to the root of the hierarchy. Formally, this results in mixture models along each path through the hierarchy, where term probabilities at inner nodes are effectively shared among multiple terminal nodes. Since the abstraction level from which each word has been generated is unknown, it is modeled as an unobserved, hidden variable. The abstraction levels of words give more intuitive theoretical model of the way words are selected to form documents on a particular topic. In addition, empirical results show practical advantages of this model for obtaining class-conditional term probabilities $P(w|c)$ for the purposes of text classification.

Our model also draws from the Cluster Abstraction Model (CAM) [4], a method for creating a hierarchical model from *unlabeled* data. In [4], the goal is the unsupervised learning of the hierarchical model. An annealed EM procedure is used for learning the hierarchy, and the phase transitions are used to identify the levels in the tree. In both CAM and our model, the hierarchy represents hierarchical relations between document groups. The generative models for documents are very similar. However, in contrast to CAM, our model is given a predefined hierarchy and labeled training data. The labeled data is used for estimating the model parameters using an EM procedure, where the structure of the model is given. The goal of our model is to *categorize* new unlabeled data. This is done by computing the posterior, $P(c|d)$, without the need for EM or folding-in. We use the generative model to obtain better class conditional word distributions $P(w|c)$ using a mixture of abstraction levels and thus achieving selective shrinkage as discussed earlier.

2.2 Hierarchical Mixture Model Description

The generative Hierarchical Mixture Model is as follows: The training data is a collection of documents. Each document is a sequence of words $d = (w_1, w_2 \dots w_{l(d)})$, where $l(d)$ denotes the document length. The document is assumed to be generated from repeated independent random trials, where the number of trials corresponds to the length of the document (and hence is fixed). Each trial can be decomposed into a two-step generative process.

- Select an abstraction level v in the hierarchy with probability $P(v|c(d))$.
- Select a word w conditioned on v with probability $P(w|v)$.

Therefore, the probability of generating word w in a document d given that d belongs to class c is given by:

$$P(w|c) = \sum_{v \uparrow c} P(v|c)P(w|v), \quad (1)$$

where the notation \uparrow refers to all inner nodes v above the terminal class node c , $c = \text{class}(d)$. The joint class-conditional probability of the words in a document is thus given by:

$$P(w_1, w_2, \dots, w_{l(d)}|c) = \prod_{j=1 \dots l(d)} \sum_{v \uparrow c} P(v|c)P(w_j|v) \quad (2)$$

The likelihood with respect to the overall document collection is then:

$$\prod_{i=1 \dots n} P(w_1, w_2, \dots, w_{l(d_i)}|c(d_i)) = \quad (3)$$

$$= \prod_{i=1 \dots n} \prod_{j=1 \dots l(d_i)} \sum_{v \uparrow c(d_i)} P(v|c(d_i))P(w_j|v) \quad (4)$$

The model parameters are the node-conditional word distributions $P(w|v)$ and the class-conditional distributions of abstraction levels $P(v|c)$. The probabilities $P(v|c)$ are set to zero for any node not on the path to the root from the node corresponding to c . In the spirit of model interpolation, we use separate held-out data for fitting the mixture weights $P(v|c)$. Moreover, tempered EM [5, 6], a generalization of EM, was used for parameter estimation. This has proven to be advantageous in practice, but we are not including detailed experimentation on the value of tempering. It has been shown before [5], that the gains of tempering can be significant. The Expectation Maximization procedure repeats the following E-step and M-step.

E-step (Expectation step):

$$P(v|c, w_j) = \frac{P(w_j|v)P(v|c)}{\sum_{v' \uparrow c} P(w_j|v')P(v'|c)}. \quad (5)$$

These posterior probabilities have to be computed for each valid (c, v) pair and for each word w_j that occurs in documents belonging to class c . In the worst case this amounts to $K \times M \times r$ posterior calculations, where K is the number of classes, M is the size of the vocabulary and r is the maximal depth of the tree.

M-step (Maximization step):

$$P(w_j|v) = \frac{\sum_{c \downarrow v} n_{jc} P(v|c, w_j)}{\sum_{w' \in W} \sum_{c \downarrow v} n_{jc} P(v|c, w')}, \quad (6)$$

where n_{jc} are the cumulative counts $n_{jc} = \sum_{d_i, c(d_i)=c} n_{ij}$, n_{ij} denotes the standard term-frequencies and W is the set of all words in the vocabulary.

The second part of the M-step deals with the mixing proportions which are estimated from held-out data with cumulative counts n'_{jc} ,

$$P(v|c) = \frac{\sum_{w_j \in W} n'_{jc} P(v|c, w_j)}{\sum_{v' \uparrow c} \sum_{w_j \in W} n'_{jc} P(v'|c, w_j)}, \quad (7)$$

The M-step thus consists of two parts – updating the node-conditional word distributions $P(w|v)$, and updating the

mixing proportions $P(v|c)$ over abstraction levels for each terminal node (class c).

As mentioned before, the mixture weights, $P(v|c)$, were set in a separate EM procedure which maximizes the likelihood of a separate held-out set. In practice, we did not use additional held-out documents for estimation of these parameters but used the same training set in a leave-one-out fashion.

The major benefit of this parameterization according to the two-step generative model is that words can move vertically, i.e., more general words become more probable at higher nodes of the tree, and class-specific words that discriminate well between classes become more probable at the leaf nodes. The Hierarchical Shrinkage model [14] fits only the interpolation weights λ , which are similar to the mixture weights $P(v|c)$ here, but treats the node-conditional term distributions at inner nodes $P(w|v)$ as fixed. Consequently, the term distributions for inner nodes do not change and are always formed by pooling data from their children. Fitting the word-distributions at each node through EM makes it possible to perform selective shrinkage, i.e. more general words come from upper levels and very specialized ones come from the leaf nodes. Empirically we found that doing more than two to five iterations of EM for reestimating the node-conditional word distributions was unnecessary and lead to overfitting. Therefore the Hierarchical Mixture model does not require significant computational resources beyond the requirements of the simpler Hierarchical Shrinkage model.

This model can be used for classification in the following way: For single category assignments, the class with the highest posterior probability given the words of the document, $P(c|d)$, is chosen. The posterior $P(c|d)$ is computed by Bayes rule :

$$P(c|d) = \frac{P(c) \prod_{j=1 \dots l(d)} P(w_j|c)}{\sum_{c'} P(c') \prod_{j=1 \dots l(d)} P(w_j|c')} \quad (8)$$

$$P(w_j|c) = \sum_{v \uparrow c} P(v|c)P(w_j|v) \quad (9)$$

For the case of overlapping categories we suggest to set a threshold for each class based on the likelihood ratio statistic $\log P(d|c) - \log P(d)$, where $P(d) = \prod_{j=1 \dots l(d)} P(w_j)$ with pooled unigram probabilities $P(w_j)$. The threshold is optimized using held-out data. Intuitively, this measures how much more likely a document is under the hypothesis that it belongs to a specific class c compared to the null hypothesis of not belonging to any class. We use the Hierarchical Shrinkage Model [14] for multi-category classification in exactly the same way after obtaining $\log P(d|c)$ and $\log P(d)$ according to that model.

3. EXPERIMENTS

3.1 Corpora

We evaluated the performance of the Hierarchical Mixture model and compared it to the performance of the Naive Bayes classifier, Probabilistic Latent Semantic Analysis, Hierarchical Shrinkage, k -NN, and SVMs on two corpora.

3.1.1 20 Newsgroups

The Newsgroups data set [10] was collected by Ken Lang and contains approximately 20,000 newsgroup postings distributed among 20 different newsgroups. Each newsgroup consists of about 1,000 postings. We experimented with only 15 of the newsgroups, to make our results comparable to the results in [14]. These are not “easier” newsgroups; some of the newsgroups are very similar in subject matter and around 4% of the texts are even cross-posted. The 15 newsgroups fall naturally into 5 upper level categories. Table 1 shows the 15 newsgroups and their organization in upper level topics. The hierarchical models were trained using this hierarchy - root plus five nodes at depth one plus fifteen nodes (leaves) at depth two. We included the subject header in the text of the documents. The documents were tokenized and all letters downcased. We did not use word stemming. The word types that occurred less than 3 times in the overall document collection were discarded. We also used a stop word list to remove some of the most frequently occurring word types. To examine the performance of the methods when trained with varying numbers of training exemplars, equal-sized subsets of documents from each newsgroup were created for training and validation, ranging in size from 7 to 677 documents. For the methods that did not require separate held out validation exemplars, the training and validation data was combined and used to train the method. A separate set of exemplars were used for testing.

Table 1: Topic Hierarchy for 15 Newsgroups

<i>COMPUTERS</i>	
comp.graphics	comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware	comp.sys.mac.hardware
comp.windows.x	
<i>MOTORS</i>	
rec.autos	rec.motorcycles
<i>POLITICS</i>	
talk.politics.guns	talk.politics.mideast
talk.politics.misc	
<i>RELIGION</i>	
alt.atheism	soc.religion.christian
talk.religion.misc	
<i>SPORTS</i>	
rec.sport.baseball	rec.sport.hockey

3.1.2 Reuters

We used the ModApte split of the Reuters-21758 corpus¹, with the modification used by Yang and Liu[18] such that only documents assigned to categories that occur in both

¹The Reuters-21578 data set was obtained at <http://www.research.att.com/lewis/reuters21578.html>

Table 2: Mid-level Clusters of Reuters Categories

<i>COMMODITIES</i>		
barley	carcass	castor-oil
cocoa	coconut	coconut-oil
coffee	copra-cake	corn
cotton	cotton-oil	grain
groundnut	groundnut-oil	hog
l-cattle	lin-oil	livestock
lumber	meal-feed	oat
oilseed	orange	palm-oil
palmkernel	pet-chem	potato
rape-oil	rapeseed	rice
rubber	rye	ship
sorghum	soy-meal	soy-oil
soybean	sugar	sun-meal
sun-oil	sunseed	tea
veg-oil	wheat	
<i>FINANCIAL</i>		
acq	bop	cpi
cpu	dfl	dlr
dmk	earn	gnp
housing	income	instal-debt
interest	ipi	jobs
lei	money-fx	money-supply
nkr	nzdlr	rand
reserves	retail	trade
wpi	yen	
<i>METALS</i>		
alum	copper	gold
iron-steel	lead	nickel
palladium	platinum	silver
strategic-metal	tin	zinc
<i>ENERGY</i>		
crude	fuel	gas
heat	jet	naphtha
nat-gas	propane	

the training and test sets are included. In this set, there were 90 categories: a training set of 7769 documents, and a test set of 3019 documents. We downcased but did not perform stop word removal, although we did dimensionality reduction, as outlined below.

Reuters assigns one or more categories to each document. Categories are also grouped into 8 category sets, but the logic behind the groupings may not be appropriate for our shrinkage application. For example, 4 (Subject, Economic Indicator, Currency, Corporate) of the 8 categories are related to various aspects of finance. We chose instead to create a different hierarchy for use in shrinkage. We represented each category as the sum of the term counts for all documents assigned to that category. Using agglomerative clustering, four major categories were identified and used as an intermediate layer in the category hierarchy. The four categories roughly corresponded to commodities, financial, metals, and energy, and there were 50, 24, 9 and 7 topics assigned to each, respectively. These categories seem to correspond well with those identified by Weigend et al. [17]. The topics clusters are shown in Table 2. Docu-

ments may be assigned to multiple categories at the lowest level of the hierarchy.

3.1.3 Dimensionality Reduction

To reduce the dimensionality of the document term frequency vectors for some of our experiments, we applied the now-standard technique of retaining only the N_t terms that are deemed individually most informative about the document class label, where $N_t \in \{1k, 2k, 10k\}$. Specifically, for each term w in the overall vocabulary, we estimated the average reduction in class uncertainty achieved by knowing whether or not w occurs in a document (denoted $w \in d$) as

$$\begin{aligned} R(w) = & H(c) - H(c \mid \text{knowledge of whether } w \in d) = \\ & - \sum_{c \in \mathcal{C}} P(c) \log P(c) \\ & + \Pr(w \notin d) \sum_{c \in \mathcal{C}} P(c \mid w \notin d) \log P(c \mid w \notin d) \\ & + \Pr(w \in d) \sum_{c \in \mathcal{C}} P(c \mid w \in d) \log P(c \mid w \in d) \end{aligned} \quad (10)$$

where all probabilities in (10) were estimated from the training set's category-term cooccurrence matrix. Doing so avoids any inconsistencies in the estimates of the marginal and conditional probabilities that might otherwise arise because of the presence of multiple category labels for each document.

3.2 Classifiers

Here we briefly outline the other classification methods used in the evaluation.

3.2.1 Naive Bayes

This model has been discussed extensively in the literature for more than forty years, cf. [11, 15] for an overview. In our experiments, we have used Naive Bayes with a multinomial sampling model (cf. [13, 16]).

The Naive Bayes method views documents as bags of words. At its heart is a crucial conditional independence assumption, namely that the probability of a word occurrence is independent of all other word occurrences in the document (the context), given the knowledge about the document's class membership. The class conditional document probability thus has the following factorial form:

$$P(w_1, w_2, \dots, w_m \mid c(d)) = \prod_{j=1 \dots m} P(w_j \mid c(d)) \quad (11)$$

where (w_1, w_2, \dots, w_m) is the sequence of words in the document and $c(d)$ is the class to which d belongs.

A new document is assigned to the class which has maximum posterior probability given the words of the docu-

ment:

$$\begin{aligned} c(d') &= \arg \max_{k=1 \dots K} P(c_k \mid w_1, w_2, \dots, w_m) \quad (12) \\ &= \arg \max_{k=1 \dots K} P(w_1, w_2, \dots, w_m \mid c_k) p(c_k) \quad (13) \end{aligned}$$

The class-conditional word distributions $P(w \mid c_k)$ are estimated based on relative frequencies extracted from training data. In our implementation we paid special attention to the probability assigned to zero frequency events. Instead of adding one to the counts of all events, a technique known as Laplace smoothing, we applied Lidstone smoothing [12] with parameters estimated from held-out data. In Lidstone smoothing, we add some usually smaller than one parameter λ to each count from training data.

We used a variation of this model for multi-label categorization. If a document had multiple labels in the training set, equal fractions of its word counts were added into the pooled word counts of each of the categories. Similarly to the hierarchical mixture model, we then set a threshold for each class based on the likelihood ratio statistic $\log P(d \mid c) - \log P(d)$, where $P(d) = \prod_{j=1 \dots l(d)} P(w_j)$. The thresholds were optimized on held-out data. We learned only one probability model for all categories, as opposed to having a separate binary classifier for each category.

3.2.2 Probabilistic Latent Semantic Analysis

This approach was proposed in [5]. It defines a generative data model. Here, a document is not generated by the single class to which the document belongs, but by a document-specific mixture of topics.

$$\begin{aligned} P(w, d) &= P(d) P(w \mid d) \\ &= P(d) \sum_{k=1 \dots K} P(w \mid c_k) P(c_k \mid d) \end{aligned} \quad (14)$$

The word-document co-occurrence events are assumed to be independent. Each occurrence of a word in a document is generated by an unobserved topic c . If we know the document specific distribution over classes $P(c \mid d)$, we can classify the document to the most likely class. Given a set of training documents with corresponding classes, we can estimate topic-specific word distributions and document-specific topic distributions. We used EM to find the maximum likelihood parameters from training data. Given a new test document, we can classify it by first doing several EM iterations to estimate its document specific topic distribution. We estimated the model parameters by first initializing the topic specific word distributions by the relative frequency estimates from training data smoothed with Lidstone smoothing with held-out estimation as above. The document-specific topic distributions were initialized as $P(c \mid d) = 1$ if d had a class c and close to 0 otherwise.

3.2.3 Hierarchical Shrinkage

We tried to replicate the Hierarchical Shrinkage model for text classification proposed in [14] as accurately as possible. The key idea in this model is to use a hierarchy

of classes and to interpolate the parameters for the class-conditional term distributions of a Naive Bayes classifier with more reliable estimates from data-rich parents in the hierarchy. This form of shrinkage effectively trades in some estimation bias for a reduction of in the estimator's variance: data that is known not to belong to a particular class c is utilized in order to estimate the class-conditional term probabilities $P(w|c)$. As was shown in [14], introducing this bias pays-off, in particular in the regime of very sparse data. Having the hierarchy of topics, we can compute maximum likelihood estimates for node-conditional word distributions at each node. The documents that belong to a class (leaf node in the tree) are also assumed to belong to each of the nodes along the path to the root. Thus the root contains all documents in the training data and its estimates are most reliable, but they are too class-unspecific. A separate node above the root is added as well, which holds the uniform distributions over words. The new estimates for class-conditional word distributions are linear combinations of the MLE estimates for the word-conditional distributions on the path from the leaf of this class to the root.

The interpolation parameters λ are obtained through maximization of the likelihood on held-out data. This model has been shown to improve the classification accuracy compared to Naive Bayes on three data sets especially on classes for which little training data was available [14].

3.2.4 k -NN

The k -nearest neighbor (k -NN) classifier is one of the top-performing classifiers for the categorization task [18]. For a given test document, the system identifies the k nearest neighbors. Based on the categories assigned to these neighbors, a score for each category is computed and the document is classified. We implemented k -NN based on the work given in [18]. In their work, binary category assignments are made using a learned category-specific threshold. In particular, for each test document, x , a score indicating whether a document belongs to category c_j is computed as:

$$y(x, c_j) = \sum_{d_i \in kNN_j} \text{sim}(x, d_i) - b_j \quad (15)$$

where d_i represents a training document, and kNN_j represents the subset of training documents in the k nearest neighbors that are labeled as class c_j . The similarity between test document x and training document d_i , $\text{sim}(x, d_i)$, was computed using a TF-IDF weighted cosine distance. The category-specific threshold, b_j , is set to the threshold that yields the best F_1 score on a validation set of documents. F_1 was calculated as $F_1(r, p) = \frac{2rp}{r+p}$, where precision p and recall r are equally weighted. In our work, rather than using a separate validation set, we employed leave-one-out and renormalized to set the category-specific thresholds. In the multi-category task, a document is assigned to category c_j if the score $y(x, c_j)$ computed in Equation 15 is positive. In the single category task, the

category with the largest value of $y(x, c_j)$ is selected. For the experiments with the Reuters dataset, we tested the method using the full term vector and term vectors containing the 1k, 2k, and 10k most informative terms, as described in Section 3.1.3. The vectors with 2k terms had the best microF1 performance, so those results are what we report in Section 4.

3.2.5 Support Vector Machines

Support-vector machine classifiers have distinguished themselves in recent years as a well-performing, computationally attractive, and theoretically rich method for text classification [3, 8]. As a representative and conveniently available implementation, we selected version 3.50 of Joachims' *SVMLight* package [8] to generate results against which to compare. To classify documents into multiple categories, a separate SVM classifier was used for each class. Except for the specification of misclassification costs, the default training parameters were used, including a linear kernel function. Unequal misclassification costs were specified in order to obtain a point on the ROC curve (which compares the true positive vs false positive rates) that allowed for better comparison with the other methods. Specifically, false negatives were weighted ten times higher than false positives during the training process.

In addition to the dimensionality reduction procedure described in Section 3.1.3, we also applied the following version of the *tf-idf* term-weighting scheme for each term in each document:

$$\text{weight} = (1 + \log tf) \log \frac{N_d}{df}$$

where N_d is the number of documents, tf is the number of times the term occurred in the document, and df is the number of documents in which the term occurred. These weights were "cosine normalized" to have unit magnitude for each document.

4. RESULTS AND DISCUSSION

Figure 1 shows the accuracy of classification on the 15 newsgroups for different sizes of data sets. The horizontal axis shows the total number of documents per class used in training, which includes training and held-out data. These results were obtained through 10-fold cross-validation, that is, the total available documents were separated into train and test splits in 10 different ways and accuracy results for the 10 splits were averaged (arithmetic mean). The vertical axis is the averaged accuracy. The figure displays the learning curves for Naive Bayes, Probabilistic Latent Semantic Analysis (PLSA), the Hierarchical Shrinkage, the Hierarchical Mixture model, and k -NN. One interesting observation is that PLSA performs very similarly to the Hierarchical Shrinkage model for all data set sizes. Naive Bayes is significantly weaker than the other three for small data sets (10-100) and all models have similar performance once the whole labeled document set is used (size 667). Figure 1 shows the learning curves for data size per class up

to about 67. The accuracy figures for all data sizes that we experimented with can be seen in Table 3.

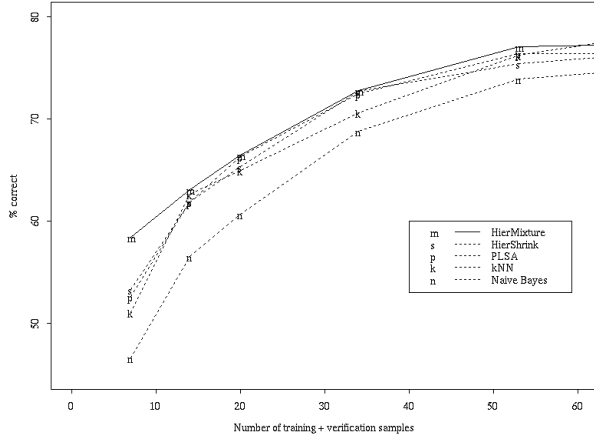


Figure 1: Performance curves for the newsgroups dataset for Naive Bayes, PLSA, Hierarchical Shrinkage and Hierarchical Mixture models.

Table 3: Performance Summary for Newsgroups

Method	7	14	20	34	48	67	133	667
NB	.467	.566	.607	.688	.739	.749	.807	.884
PLSA	.527	.619	.663	.725	.764	.764	.813	.886
HS	.533	.618	.654	.727	.754	.763	.814	.888
kNN	.511	.627	.650	.706	.762	.780	.824	.879
HM	.585	.631	.665	.728	.771	.773	.814	.888

Table 4: Performance Summary for Reuters

Method	miP	miR	miF1	maP	maR	maF1	acc
HM	.872	.771	.818	.815	.408	.459	.9953
SVM	.881	.868	.875	.837	.524	.554	.9966
kNN	.790	.813	.802	.759	.503	.491	.9938
HS	.849	.780	.813	.765	.392	.432	.9950
NB	.825	.713	.765	.763	.283	.337	.9940

When training data is extremely sparse (7 documents in Figure 1 for training and validation per class), the Hierarchical Mixture model has 11% error reduction compared to the second-best performing model (Hierarchical Shrinkage), and 22% error reduction as compared to Naive Bayes. It can be noted in Table 3 that the advantage of the Hierarchical Mixture model is less striking as training data increases, and all methods perform similarly when there is a sizable amount of training data.

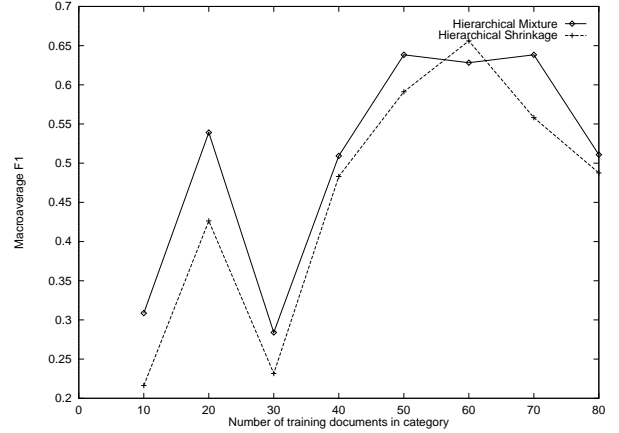


Figure 2: Performance difference between HS and HM for Reuters categories with a small number of training documents.

For reference, Table 4 shows the categorization performance results for the HM, SVM, k -NN, HS, and NB classifiers on the Reuters data when *all* of the available labeled data is used for training. The performance was measured as micro-averaged precision (miP), micro-averaged recall (miR), micro-averaged F1 (miF1), macro-averaged precision (maP), macro-averaged recall (maR), and macro-averaged F1 (maF1)[12]. The SVM technique performs well; in fact better than that reported by [18], who reported miF1= .8599 and maF1=.5251. More work is required to assess how its performance responds when the amount of available data becomes severely limited. Our results for kNN and NB are somewhat lower than that published by [18] (miF1/maF1 of .8567/.5242 and .7956/.3886 for kNN and NB, respectively). We hypothesize that the differences in our results are due to differences in data preparation, although we did not find details on data preparation in [18]. For example, we did not perform stemming, the stop word lists are probably different, and different term weighting schemes may have been used.

We also examined the difference in performance between the Hierarchical Shrinkage and the Hierarchical Mixture models on categories with a small number of positive examples. This is shown in Figure 2 for the Reuters categories, which have varying numbers of training documents. The horizontal axis is number of training documents in the category and the vertical axis is macro-averaged F1 on categories with this number of training samples. For example, at 10 on the horizontal axis, the macro-averaged F1 of the two classifiers on all categories with less than or equal to 10 training documents is shown. At 20 on the horizontal axis, the macro-average F1 on categories with more than 10 and less than 20 training documents, etc. is shown. Even though the Hierarchical Shrinkage model is using the topic hierarchy to provide better estimates for categories with a small number of labeled documents, the

Hierarchical Mixture model achieves better results by also fitting the node-conditional word distributions and differentiating better among class-specific and general terms.

5. SUMMARY

We have presented a method for text categorization that effectively makes use of hierarchical topic structure to increase performance when the amount of labeled data is limited. This is achieved through the use of selective shrinkage to provide better estimates of the class-conditional word distributions, which was found to be particularly effective when the amount of labeled data is small, e.g., around 7 documents per category. We evaluated the method on single category and multi-category categorization tasks on the Newsgroups and Reuters-21578 datasets, respectively, comparing the method against Naive Bayes, hierarchical shrinkage, PLSA, k -NN, and SVM models. For the Newsgroups experiments, we varied the amount of labeled documents used for training. When a large number of labeled documents was available, all methods were found to perform quite well. In our Newsgroups experiments involving the practically important situation of severely limited labeled data, the hierarchical mixture model appears to offer distinct performance advantages over the other techniques considered. For the experiments with the Reuters corpus, we have presented a method for multi-category classification and showed that it can outperform Hierarchical Shrinkage and Naive Bayes.

6. REFERENCES

- [1] S. D'Alessio, M. Murray, R. Schiaffino, and A. Kershenbaum. Category levels in hierarchical text categorization. In *Proceedings of EMNLP-3, 3rd Conference on Empirical Methods in Natural Language Processing*, 1998.
- [2] S. Dumais and H. Chen. Hierarchical classification of web content. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 256–263, Athens, Greece, August 2000.
- [3] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 148–155, 1998.
- [4] T. Hofmann. The cluster-abstraction model: unsupervised learning of topic hierarchies from text data. In *Proceedings of the International Joint Conference in Artificial Intelligence*, pages 682–687, 1999.
- [5] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 50–57, Berkeley, California, August 1999.
- [6] T. Hofmann and J. Puzicha. Statistical models for cooccurrence data. AI-MEMO 1625, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1998.
- [7] F. Jelinek and R. Mercer. Interpolated estimation of Markov source parameters from sparse data. In S. Gelsema and L. Kanal, editors, *Pattern Recognition in Practice*, pages 381–402. North-Holland, 1980.
- [8] T. Joachims. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.
- [9] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the International Conference on Machine Learning*, pages 170–178, 1997.
- [10] K. Lang. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*, pages 331–339, 1995.
- [11] D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the 1998 European Conference on Machine Learning*, 1998.
- [12] C. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2000.
- [13] A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.
- [14] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 359–367, Madison, Wisconsin, 1998.
- [15] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [16] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.
- [17] A. Weigend, E. Wiener, and J. Pedersen. Exploiting hierarchy in text categorization. *Information Retrieval*, pages 193–216, 1999.
- [18] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 42–49, Berkeley, August 1999.