In *Proceedings of the Fourteenth International Conference on Machine Learning (ML-97)*, pages 170-178, Nashville, Tennessee, July 1997

Hierarchically classifying documents using very few words

Daphne Koller
Gates Building 1A
Computer Science Department
Stanford University
Stanford, CA 94305-9010
koller@cs.stanford.edu

Mehran Sahami
Gates Building 1A
Computer Science Department
Stanford University
Stanford, CA 94305-9010
sahami@cs.stanford.edu

Abstract

The proliferation of topic hierarchies for text documents has resulted in a need for tools that automatically classify new documents within such hierarchies. Existing classification schemes which ignore the hierarchical structure and treat the topics as separate classes are often inadequate in text classification where the there is a large number of classes and a huge number of relevant features needed to distinguish between them. We propose an approach that utilizes the hierarchical topic structure to decompose the classification task into a set of simpler problems, one at each node in the classification tree. As we show, each of these smaller problems can be solved accurately by focusing only on a very small set of features, those relevant to the task at hand. This set of relevant features varies widely throughout the hierarchy, so that, while the overall relevant feature set may be large, each classifier only examines a small subset. The use of reduced feature sets allows us to utilize more complex (probabilistic) models, without encountering many of the standard computational and robustness difficulties.

1 Introduction

Over the past decade, we have witnessed an explosion in the availability of online information, with millions of documents on every topic easily accessible via the Internet. As the available information increases, the inability of people to assimilate and profitably utilize such large amounts of information becomes more and more apparent. The most successful paradigm for organizing this mass of information, making it comprehensible to people, is by categorizing the different documents according to their topic, where topics are organized in a hierarchy of increasing specificity.

Hierarchical classifications of this type have long been used in special-purpose collections of documents such as MEDLINE (Hersh *et al.* 1994) or collections of patent documents (Self 1996). More recently, they have been used in several internet search engines, such as Yahoo (Yahoo! 1995) or Infoseek (Infoseek 1995), to categorize the contents of the World Wide Web.

The bottleneck in these classification tasks is the need for a person to read each document and decide on its appropriate place in the hierarchy. Clearly, we would like to avoid this bottleneck by automatically classifying new documents. Indeed, Infoseek has recently attempted to overcome this difficulty by using neural network technology to automatically categorize webpages (Infoseek 1996). In many ways, this task is ideally suited to the application of machine learning techniques. We have a specified set of classes, i.e., the topics in the hierarchy, and a very large training set, consisting of all of the documents that have already been classified. However, with few exceptions (notably (Almuallim, Akiba, & Kaneda 1996) which focused on hierarchically structured attributes rather than classes), most work in classification has ignored the problem of supervised learning in the presence of hierarchically structured classes. (There has been some work on unsupervised hierarchical clustering, e.g., (Fisher 1987).)

Of course, standard classification techniques can be applied to this problem almost directly. We simply construct a "flattened" class space, with one class for every leaf in the hierarchy. We use the presence or absence of different words as our features. We can now train a single classifier so that each document is classified as belonging to precisely one of the possible basic classes. Unfortunately, this simplistic approach breaks down in the context of text classification. Here, the resulting classification problem is huge: for a large corpus, we may have hundreds of classes and thousands of features. The computational cost of training a classifier for a problem of this size is prohibitive. Furthermore, the variance of the resulting classifier is typically very large, since such a model will have many thousand parameters which need to be estimated, and thus can easily lead to overfitting of the training data. As a result, we are typically able to use only very simple classifiers such as Naive Bayes (Good 1965).

Previous work (Schutze, Hull, & Pedersen 1995), has shown that feature selection can be a useful tool in dealing with these issues. We eliminate many of the words that appear in the corpus as being unindicative of any topic. In previous work (Koller & Sahami 1996), we showed that one can obtain a significant increase in accuracy by reducing the number of words used for classification from 1600 to as few as 600. However, even for 600 features, the computational cost and the robustness still pose significant limitations.

In this paper, we propose a new approach to classification using a structured hierarchy of topics. Rather than ignoring the topical structure and building a single huge classifier for the entire task, we use the structure to break the problem up into manageable size pieces. The basic insight supporting our approach is that topics that are close to each other in the hierarchy typically have a lot more in common with each other than topics that are far apart. Therefore, even when it is difficult to find the precise topic of a document, e.g., color printers, it may be easy to decide whether it is about "agriculture" or about "computers".

Building on this intuition, our approach divides the classification task into a set of smaller classification problems corresponding to the splits in the classification hierarchy. Thus, for example, we may have one classifier which distinguishes articles about agriculture from articles about computers, and another one, only applied to documents about agriculture, which distinguishes animal husbandry from crop farming. Each of these subtasks is significantly simpler than the original task, since the classifier at a node in the hierarchy need only distinguish between a small number of categories. Therefore, it is possible to make this determination based only on a small set of features. For example, there appears to be a fairly small number of words—e.g., computer, farm, plant, software, ...—whose presence or absence in the document clearly differentiates documents about agriculture from documents about computers. The ability to restrict to a very small feature set avoids many of the difficulties we describe above. The resulting models are more robust, and less subject to overfitting.

It is important to note that the key here is not merely the use of feature selection, but its integration with the hierarchical structure. To understand this integration, observe that the set of features required for these subtasks varies widely from one to the other. For example, almost none of the words that can help us differentiate between agriculture and computers are useful for distinguishing between animal husbandry and crop farming: a word such as "farm" is unlikely to be helpful because it is fairly likely to appear in documents of both types, whereas a word such as "computer" is not helpful because it is likely to appear in virtually no documents that reach this classifier. Thus, while each classifier uses only a very small set of features, the overall set of features used in the classification process is still rather large. A single flattened classifier would have to consider all of these features in order to do a reasonable job of classifying all of the documents. For any given document, however, most of these features are irrelevant, and serve only to confuse the classifier. In the hierarchical approach, a document percolating down the hierarchy of classifiers only encounters questions concerning a small fraction of the features throughout the process (e.g., a document about computers will probably never meet a classifier utilizing the word "cow"). Even the features which each classifier does utilize are divided so as to focus the attention of each classifier on the features relevant to the classification subtask at hand.

The reduction in the feature space also allows us to go beyond the simple classifiers such as Naive Bayes, to which we are restricted in tasks involving a very large number of features. For example, we can train a probabilistic classifier, such as TAN (Friedman & Goldszmidt 1996a) or KDB (Sahami 1996), that takes into account the correlation between different features (e.g., the fact that *Microsoft* and *Windows* tend to cooccur). Such classifiers provide a more realistic model for text data, potentially leading to higher classification accuracy. However, the search in this more complex hypothesis space is significantly more expensive, being at least quadratic (and sometimes exponential) in the number of features, versus the linear time behavior of Naive Bayes. Furthermore, even when we are willing to spend the time, these more complex models are usually unsuitable for large feature spaces, due to the problem of overfitting.

As in the case of feature selection, it is the integration of this idea with the hierarchical structure which is the key to its success. As our results show, a flat classifier is virtually incapable of taking advantage of the richer models, regardless of the size of the feature space. By contrast, when we use more expressive classifiers in the nodes of our hierarchy, the classification accuracy increases substantially. We conjecture that, as for the choice of feature set, the dependency models appropriate for the different nodes in the hierarchy are radically different. In the flat classification case, the dependency model would be an aggregate of the various lower-level models. As a consequence, the dependencies either "wash out" or are too complex to represent using a limited dependency model.

The use of the hierarchical structure allows us to focus both our feature space and the dependency model on the relevant distinctions. As we show, the combination of these three techniques provides significant accuracy gains over the standard flat approach.

We note that, while our techniques are designed for dealing with the huge classification tasks arising in the context of text classification, they may also be useful in other domains. For example, in medical applications, we often want to classify a patient's disease based on symptoms and test results. Here also, our classes—the diseases—are often organized in a taxonomic hierarchy, where only a small number of features is needed to distinguish between neighboring classes.

The rest of this paper is structured as follows. In Section 2 we discuss the specific techniques we use for feature selection and for classification. We focus on probabilistic techniques, as they provide a coherent underlying framework both for feature selection and for construction of classifiers of various complexities. We emphasize, however, that our basic paradigm in no way depends on the use of these particular techniques. In Section 3 and Section 4, we provide our experimental methodology and a variety of results supporting our approach. We show that our technique allows us to restrict the set of features significantly (from over 1000 to 10), and that the resulting classifier, in addition to being smaller and easier to train, also provides better accuracy than the flat classifier. We conclude in Section 5 with some discussion and directions for future work.

2 Probabilistic Framework

Our general approach, as described in the introduction, consists of constructing a hierarchical set of classifiers, each based on its own set of relevant features. It uses two main subroutines: a feature selection algorithm for deciding on the appropriate feature set at each decision point, and a supervised learning algorithm for constructing a classifier for that decision. The general approach can be instantiated in a variety of ways, depending on the choice of these subroutines.

In this paper, we have chosen to focus on probabilistic methods for feature selection and for classification. The probabilistic framework provides both efficient and principled techniques for pruning large feature sets (Koller & Sahami 1996) and a range of classifiers of varying complexities and accuracies (Pazzani 1995; Friedman & Goldszmidt 1996a; Sahami 1996; Singh & Provan 1996). We now provide a brief overview of the probabilistic framework and its application to classification and feature selection.

2.1 Bayesian Classifiers

At the heart of the probabilistic framework is the idea that our model of the world is represented as a probability distribution over the space of possible states of the world. Typically, a state of the world is described via some set of random variables, so that each such state is an assignment of values to these variables.

A Bayesian network (Pearl 1988) allows us to provide compact descriptions of complex distributions over a large number of random variables. It uses a directed acyclic graph to encode *conditional independence* assumptions about the domain; these independence assumptions allow the distribution to be described as a product of small *local interaction models.* Each variable (feature) X_i is represented as a node in the network. An arc between two nodes denotes the existence of a direct probabilistic dependency between the two variables. Essentially, the structure of the network denotes the assumption that each node X_i in the network is conditionally independent of its non-descendants given its parents $\Pi(X_i)$. To describe a probability distribution satisfying these assumptions, we associate with each node X_i in the network a *conditional probability table*, which specifies the distribution over X_i given any possible assignment of values to its parents $\Pi(X_i)$. If X_i has no parents, it simply contains a prior probability distribution over X_i's values. The network structure and the associated parameters uniquely define a probability distribution over the variables in the network.

A Bayesian classifier is simply a Bayesian network applied to a classification domain. It contains a node C for the (unobservable) class variable and a node X_i for each of the features. Given a specific instance $\mathbf{x}$ (an assignment of values $x_1, x_2, ..., x_n$ to the feature variables), the Bayesian network allows us to compute the probability $P(C = c_k \mid \mathbf{X} = \mathbf{x})$ for each possible class c_k. *Bayes Optimal* classification can be achieved by simply selecting the class c_k for which this probability is maximized.

While it is possible to use any Bayesian network over these variables as a Bayesian classifier (Singh & Provan 1996), empirical evidence (Friedman & Goldszmidt 1996a) suggests that networks where the feature variables are all directly connected to the class variable are better at the classification task. The simplest and earliest such classifier is the Naive Bayesian classifier (Good 1965). This classifier, which significantly predates the development of Bayesian networks, is still widely employed today. The Naive Bayesian classifier makes the simplifying, but very restrictive, assumption that domain features are conditionally independent of one another, given the class variable. In other words: $P(\mathbf{X}|C) = \prod_i P(X_i|C)$. This assumption corresponds to the Bayesian network structure of Figure 1(a).

The assumption that all features are conditionally independent given the class variable is clearly unrealistic in the text domain as well as in others. Several approaches have been proposed for augmenting the Naive Bayesian classifier with limited interactions between the feature variables, i.e., where we allow each node to have some parents beyond the class variable (as illustrated in Figure 1(b)). Unfortunately, the problem of inducing an optimal Bayesian classifier is NP-hard even if we restrict each node to have at most two additional parents (Chickering 1995). Thus, any optimal algorithm for constructing such a classifier would be exponential in the number of features in the worst case.

Two main solutions have been proposed to this problem: The TAN algorithm of (Friedman & Goldszmidt 1996a) restricts each node to have at most one additional parent, in which case an optimal classifier can be found in quadratic time (in the number of features). The KDB algorithm of (Sahami 1996), on the other hand, compromises by heuristically searching for a good, but potentially suboptimal, structure. It can be used for finding classifiers where each node has at most k parents for arbitrary values of k. Essentially, it chooses as the parents of a node X_i the k other features that X_i is most dependent on, using a metric of class conditional mutual information, $I(X_i; X_j|C)$ (Cover & Thomas 1991). Since part of our goal in this paper was to experiment with classifiers of varying complexity, we chose to use KDB as the basis for our experiments.

The structure selection phase is done in a greedy fashion, requiring time which is linear in k and quadratic in the total number of features. (Of course, the size of the conditional probability table for a node with k parents is exponential in k, and requires a corresponding amount of time for the parameter estimation phase.)

2.2 Feature Selection

Recall that in our text domains, we have a feature for every word that appears in any document in the corpus. Even if we use algorithms such as KDB or TAN, which are "merely" quadratic (as opposed to exponential) in the total number of features, the cost can still be prohibitive. Moreover, if we wish to employ more optimal models which are of exponential complexity in the size of the feature set, then feature selection is an absolute must. For text domains in particular, the number of *a priori* features is overwhelming so that feature selection is imperative (even if we ignore the improved classification benefits of hierarchically-applied feature selection, as outlined in the introduction).

In addressing this issue, we continue in the probabilistic framework, applying the feature selection method of Koller & Sahami (1996). Essentially, the algorithm greedily eliminates features one by one so as to least disrupt the original conditional class distribution. For each remaining feature X_i, the algorithm finds a feature set MB_i which approximates the *Markov Blanket* of X_i, i.e., the set of features that render X_i conditionally independent of the remaining domain features. The algorithm then determines the expected cross-entropy (Cover & Thomas 1991) $\delta_i = P(X_i, MB_i) D(P(C|X_i, MB_i), P(C|MB_i))$, where $D(\mu, \sigma) = \sum_{x \in \Omega} \mu(x) \log \frac{\mu(x)}{\sigma(x)}$, is the cross-entropy between two distributions μ and σ over the same probability space Ω. The algorithm then eliminates the feature X_i for which δ_i is minimized. This process can be iterated to eliminate as many features as desired. To compute $P(C|X_i, MB_i)$ the feature selection algorithm simply uses the Naive Bayes model, assuming $MB_i = \emptyset$, and can thus run quickly even on thousands of features. In this respect, the algorithm is very applicable to text domains with many features. Previously, we have demonstrated this in experiments with text, and thus this method was selected for use here.

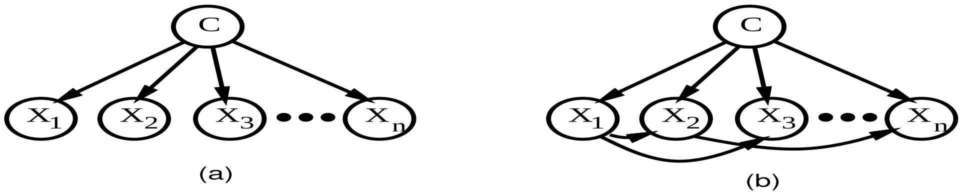

Figure 1: Bayesian networks corresponding to (a) a Naive Bayesian classifier; (b) A more complex Bayesian classifier allowing limited dependencies between the features.

3 Experimental Methodology

In order to test our scheme for hierarchical classification, we first needed to obtain hierarchically classified text data. As the basis, we used the Reuters-22173 dataset.[1] The Reuters collection does not have a predetermined hierarchical classification scheme, but each document can have multiple labels. We therefore identified labels which tended to subsume other labels, and used those as the higher level topics in our hierarchy. Three hierarchical subsets of the Reuters collection, which we call Hier1, Hier2 and Hier3, were then extracted; these datasets are described in Figure 2.

Initially, we applied a single pass of a *Zipf's Law*-based feature selection method, which eliminates all words which appear fewer than 10 or more than 1000 times in each corpus. This is done so that we do not get unrealistic improvements in accuracy by simply eliminating features that rarely if ever appear during testing, or are so frequent that they will have no bearing on classification. Previous work in information retrieval (van Rijsbergen 1979) supports the contention that such words are generally not able to improve classification accuracy.

Each document is then represented as a boolean vector, in which each feature denotes the presence or absence of a word that appeared in the corpus and survived the initial Zipf's Law-based feature selection. Thus, the number of features reported for each dataset is *after* the application of this initial feature selection. These datasets were then used in our experiments, detailed below, employing 10-fold cross-validation to produce multiple training and testing sets for each dataset.

In our experimental work, we seek to show that the hierarchical approach compares favorably with the simple approach of constructing a single large classifier over a flattened topic space. In both cases, the feature selection phase plays a crucial role in the performance of the resulting classifier.

The hierarchical classification scheme begins by applying probabilistic feature selection to the entire training dataset, using, at first, just the topics of the first-tier in the hierarchy associated with each document as classes. The resulting reduced feature set is then used to build a probabilistic classifier for the first tier of the hierarchy. We currently employ Naive Bayes and KDB with $k = 1$ and 2 as our classification methods. Then, for the training documents in each of the first-tier topics, the second-tier topics are used as class labels. For each first-tier topic, a separate round of probabilistic feature selection is employed. Note that this feature selection is done starting from the original feature set (pruned only with Zipf's law), since, as we have observed, the most indicative features at one level of the hierarchy are unlikely to be particularly useful at lower levels. Finally, we construct a separate classifier for the subtopics of each first-tier topic on the appropriate reduced feature set. Note that, since every node in the hierarchy has only a subset of the total class labels, and the nodes at the second tier of the hierarchy have fewer instances each, the additional cost of feature selection and induction is not substantially more than that of the flat classification scheme.

Test documents are classified in this hierarchy by filtering them through the first level classifier and then sending the document down to the chosen second level where a final class assignment (into a leaf node) is made. Note that errors made at the first level of the hierarchy are unrecoverable at the second level. Thus, our method needs to make two correct classifications in order for a test document to be considered properly classified. We have conducted some initial experiments in which documents were sent down multiple paths in the hierarchy of classifiers as a way of addressing this issue, but that work is beyond the scope of this paper.

In the flat classification scheme, we simply treat every

[1] This collection can be obtained by anonymous ftp from /pub/reuters1 on ciir-ftp.cs.umass.edu. Arrangements for access were made by David Lewis.

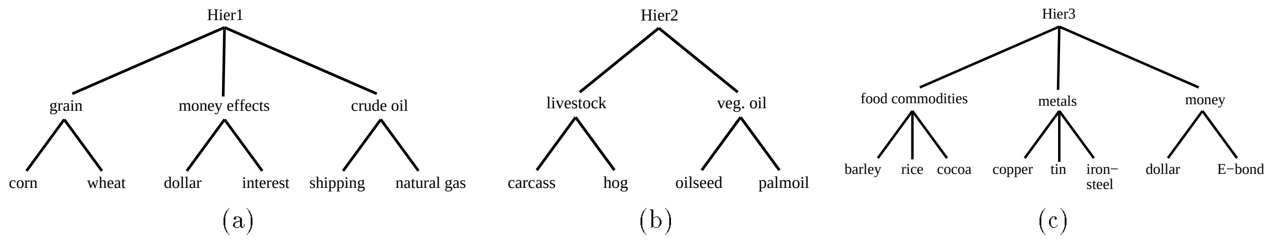

Figure 2: (a) Hier1 hierarchy—1568 features, 939 documents. (b) Hier2 hierarchy—435 features, 138 documents. (c) Hier3 hierarchy—1440 features, 834 documents.

low-level topic (leaf node) as a separate class. We then apply probabilisitic feature selection and induce one probabilistic model based on this reduced feature set.

4 Results

Since it is our belief that a small set of features suffices for accurately distinguishing between topics (and furthermore helps avoid overfitting), we employed a very aggressive feature selection policy. In the Hier1 and Hier2 domains we considered reducing the feature space to 10, 40, and 160 features. For Hier3 we considered 20, 80, and 320 features since this dataset contains the largest number of final classes among which we need to distinguish. We chose this particular pattern of feature space sizes in order to facilitate a fair comparison between the flat method and the hierarchical method. Recall that, in the hierarchical case, a potentially very different set of features is selected at each node in the hierarchy. Therefore, the hierarchical method, as a whole, actually examines a larger set of features. To allow a fair comparison, we also compare the hierarchical method with some number of features to a flat method with four times as many features (since our hierarchies contain around four classifiers each). We also considered running each method on the full feature set, but since the number of features was very large, it became prohibitive to consider models which did not assume conditional independence of features (e.g., KDB) and thus only the results for Naive Bayes are given in these cases. The accuracies and standard deviations for 10-fold cross validation are given in Table 4.

We begin by noting the substantial improvements in accuracy when feature selection was aggressively employed versus the case where all domain features were used. We see an improvement both in the hierarchical case and in the flat case, and for every single dataset. In the Hier1 dataset, these gains are particularly visible, with a statistically significant difference (T-test with $P < 0.01$) obtained in Naive Bayes for both the flat classifier with 40 features (versus the original 1568) and the hierarchical classifier with 10 features. In both cases, around 1500 features—over 95% of the features *after* Zipf's Law pruning—were eliminated! Of course, the number of features cannot be reduced to zero. In general, we observe an initial significant improvement as the number of features is reduced, and then a decrease in accuracy as the number of features is reduced too much. This phenomenon is an instance of the familiar bias-variance tradeoff. The selection of very few features, say 10, helps reduce the error associated with the variance of the model, but strongly biases the model. Conversely, the use of too many features, say 160, can cause the probability estimates used in the classification models to become very inaccurate and thus lead to poorer overall performance. We believe that combining the hierarchical method with relatively aggressive feature selection can help address this issue since it allows a larger space of *overall* features to be considered, but provides a variance control for each classifier in the hierarchy by having it focus on just a few relevant features.

We now turn our attention to the main question: the difference between the hierarchical and flat classification methods. We begin by comparing the two approaches with an equivalent number of features for each classifier. (E.g., we compare the hierarchical classifier with 10 features in each node with the flat classifier with 10 features total.) In the case of Naive Bayes, the comparison is inconclusive. In some cases (Hier1 with 10 features) the hierarchical approach is significantly better, while in others (Hier3 with 80 features) the flat approach wins. Alternatively, we can compare the cases where both classification schemes see a roughly comparable number of features (i.e., hierarchical with 10 features per node versus flat with 40 features total). In this case, the comparison for Naive Bayes is still predominantly inconclusive (with one exception in Hier3 where the flat classifier wins).

Dataset	# Features	Hierarchical NB	Hierarchical KDB-1	Hierarchical KDB-2	Flat NB	Flat KDB-1	Flat KDB-2
Hier1	10	92.4 ± 2.4	**92.4 ± 2.3**	92.2 ± 2.5	**79.3 ± 9.1**	79.1 ± 8.8	78.8 ± 8.7
Hier1	40	91.8 ± 2.8	93.3 ± 1.9	**94.1 ± 2.7**	92.0 ± 3.1	**94.1 ± 2.3**	93.8 ± 2.6
Hier1	160	88.2 ± 2.3	**91.2 ± 2.8**	91.1 ± 3.0	92.0 ± 4.1	**93.8 ± 2.8**	93.8 ± 2.8
Hier1	1568	85.9 ± 3.1	—	—	86.2 ± 2.8	—	—
Hier2	10	**87.7 ± 7.4**	85.4 ± 8.5	85.4 ± 8.5	**89.2 ± 8.3**	89.2 ± 8.3	86.9 ± 8.9
Hier2	40	88.5 ± 9.1	87.7 ± 8.3	**90.0 ± 8.2**	**87.7 ± 5.4**	86.2 ± 6.1	86.2 ± 7.1
Hier2	160	86.2 ± 10.8	**86.2 ± 10.1**	82.3 ± 10.9	**85.4 ± 8.5**	82.3 ± 12.6	77.7 ± 9.2
Hier2	435	84.6 ± 9.6	—	—	83.1 ± 8.7	—	—
Hier3	20	90.1 ± 4.1	94.7 ± 3.0	**94.9 ± 2.6**	**90.6 ± 3.4**	89.8 ± 2.4	89.8 ± 2.7
Hier3	80	90.9 ± 2.4	**98.6 ± 1.4**	97.7 ± 1.8	95.4 ± 1.7	**95.7 ± 1.5**	95.2 ± 2.7
Hier3	320	94.8 ± 2.5	**95.5 ± 2.0**	94.2 ± 2.5	**91.8 ± 4.1**	91.2 ± 2.7	89.3 ± 3.4
Hier3	1440	92.8 ± 2.5	—	—	90.9 ± 2.5	—	—

Table 1: Accuracy percentages for hierarchical and flat learning employing feature selection.

Thus, the hierarchical approach appears to present few benefits when we restrict attention to simple classifiers such as Naive Bayes. However, as we explained in the introduction, one of the primary benefits of the hierarchical approach is our ability to train more complex classifiers with richer dependency models. As we have seen, the complexity of algorithms for learning expressive classifiers grows rapidly with the number of features. Even when the growth is quadratic, as in the KDB algorithm, it is significantly more expensive to learn a single classifier over 160 features than to learn a few classifiers over only 40 features each. Furthermore, if we wish to construct even more accurate models by using an optimal Bayesian network learning algorithm, this task may be achievable in the case of 40 features, but is clearly infeasible in the case of 160.

Indeed, when we use these richer dependency models within the hierarchical approach, we begin to see significant accuracy gains over Naive Bayes. In Hier1 with 40 and with 160 features, for example, KDB2 provides an accuracy improvement which is statistically significant with $P < 0.1$. In Hier3 with 80 features, the gains are even more dramatic, with KDB1 providing an 80% reduction in error (with significance $P < 0.01$).

By contrast, the flat approach seems virtually incapable of taking advantage of the richer model space. In very few cases did we observe any improvement in accuracy in the KDB classifiers, and in no case was the improvement statistically significant. As mentioned previously, we conjecture that the reason for this shortcoming arises from the fact that the dependency models for the different classes are quite different. (This phenomenon is a form of *context-specific independence* (Friedman & Goldszmidt 1996b).) The flat classifier is required to capture, within a single model, a complex dependency structure resulting from aggregating all of these disparate dependency structures. In this case, the dependencies are either "washed out" by the noise, or are so complex that it is impossible to capture them within the restricted dependency structure that we consider.

Topic	10 most discriminating words
Top level	dollar, dealer, tonnes, agriculture, oil, grain, wheat, corn, gas, usda
Grain	london, taiwan, wheat, gulf, maize, k, corn, eep, enhancement, winter
Money Effects	dollar, japan, yen, money, england, repurchase, k, stg, shortage, system
Crude oil	production, ship, gas, natural, iran, cubic, barrel, iranian, attack, tanker

Table 2: The 10 most discriminating words in one fold of the hierarchical method for the Hier1 dataset.

Thus, we see that the key to the success of our approach is the combination of three techniques: hierarchical classification based on the structured topic hierarchy, aggressive feature selection at each node of the hierarchy, and the use of richer dependency models. The use of the hierarchy serves to focus on distributions with more uniform characteristics, allowing us to target both the selected features and the dependency model *to the local classification task.*

To illustrate this phenomenon, Table 2 shows the set of 10 features (words) found to be most dicriminating at each level of the hierarchy learned during the run of one fold on the Hier1 dataset. At the top level of the hierarchy, we see a selection of high-level terms from the various major topics. Some of these are no

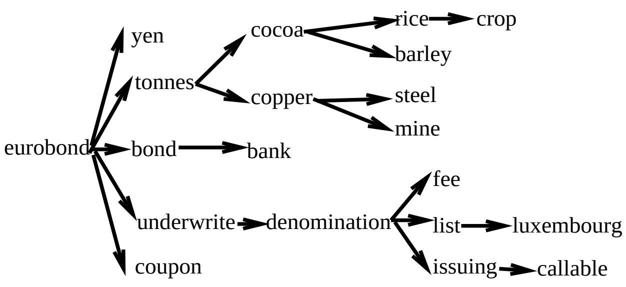

Figure 3: Word dependencies found by KDB-1 among the 20 features selected in the top level of the Hier3 hierarchy.

longer indicative at the lower levels. Thus, for example while the terms "agriculture" and "usda" are useful for identifying documents in the Grain topic, they are not useful for distinguishing among its subtopics. Rather, we see more specific words (such as "corn", "maize", and "wheat") that help distinguish between the two subtopics (Corn and Wheat) of the Grain topic. Similarly, the the Money Effects topic contains terms that help distinguish documents about the Dollar (many of which relate to "japan" and the "yen") vs. articles that relate to Interest Rates. Finally, the feature selection for the Crude Oil topic autonomously homed in on all of the terms appearing in the names of its various subtopics ("natural", "gas", and "ship"). We also note that the features selected in the different cross-validation folds contain many of the same words at each node, thereby indicating that the method is generally robust in terms of selecting meaningful features across different partitions of the same dataset.

The same localization phenomenon allows the dependency model to be tailored to the relevant distribution. To illustrate this point, we examined the actual dependencies constructed (automatically) by the KDB algorithm in the top level node in one of the folds of the Hier3 dataset (where we obtained the biggest gains from the use of richer dependency models). Recall that the task at that node was to distinguish the topics *food commodities*, *metals*, and *money*. Figure 3 shows these dependencies (omitting the universal dependence on the class variable). We see that the word dependencies form small clusters, reflecting the correlations between the words in the domain. The cluster associated with the word "tonnes" is most intriguing, as the word is used for both metals and food commodities. The model constructs two subclusters connected to "tonnes", one related to metals and the other to food commodities. This dependency helps the classifier interpret the word "tonnes" as appropriate to the context.

Our approach allows us to utilize the synergy between these tools, resulting in significantly improved classification accuracy. To see this, compare the best hierarchical classifier versus the best flat classifier for each number of features (marked in boldface in Table 4). In Hier1 and Hier2, we typically see small improvements, but not significant ones. We note that the hierarchical approach almost never detracts from accuracy, even though the hierarchical method requires the data to be fragmented at lower levels in the hierarchy. To examine the robustness of the hierarchical approach to fragmentation, we included the data impoverished Hier2 dataset. As we see, even in this case, the hierarchical method never performed significantly worse than the flat method. In fact, it actually achieved the best results on this dataset (using KDB-2 with 40 features per node) although the high variance precluded any significant accuracy results.

It is in the Hier3 dataset, which provided the largest hierarchical structure and thus the most classification information to leverage, that we see the true power of our approach. In comparing cases with equivalent numbers of features per classifier, we see that the hierarchical method significantly outperforms ($P < 0.05$) the flat classification scheme in *every* such comparison! We also compare the cases where an equivalent number of total features is used. The hierarchical method using 20 features per internal KDB classifer performs equivalently to the flat method using 80 features, but is far faster to train. Still more compelling is the fact the hierarchical method utilizing 80 features per classifier significantly outperforms ($P < 0.01$) the flat method using 320 features. Finally, we note that the hierarchical method (using KDB-1 and 80 features) achieves the highest overall classification accuracy on the Hier3 dataset, significantly outperforming *every* run of the flat method regardless of classifier or number of features used!

5 Conclusions

The recent proliferation of systems that hierarchically organize massive amounts of text-based documents calls for algorithms that hierarchically categorize new documents as they come in. We describe an approach which utilizes the existing rich hierarchical structure in order to facilitate this process. Rather than building a single massive classifier, our approach generates a hierarchy of classifiers, utilizing feature selection to tailor the feature set of each classifier to its task. As we have shown, the resulting reduction in the size of the classifier allows us to obtain significantly higher accuracy, a reduction due both to increased robustness and to our ability to use richer (and more complex) classifiers.

In future work, we hope to pursue the use of more expressive (and computationally more expensive) classifiers at the nodes of the hierarchy. In this way we hope to be able to obtain not only better classification results, but also be able to handle text collections with a wider variety of statistical characteristics.

We would also like to investigate several problems that are specific to the use of a hierarchy of classifiers. In particular, we have already mentioned the problem of recovering from classification errors early in the hierarchy. We would also like to investigate the problem of discovering new classes in the hierarchy, when we have multiple documents that don't "fit in" nicely.

Most importantly, we intend to investigate the issue of scalability by applying this method to a wider variety of text datasets. We have already conducted some very preliminary work on applying this method to small hierarchies of topics extracted from the Yahoo! web directory. These results are still inconclusive (as web data has a tendency to be extremely varied as well as incorporating other media aside from text), but we are encouraged by some of our initial results and are seeking ways to improve them. In particular, we hope to integrate such a classification method into a larger information retrieval system, thereby making use of existing subject hierarchies such as commercial Web directories.

Acknowledgements

The authors have benefitted from discussions with Adam Grove, Marti Hearst, Tom Mitchell, and Nils Nilsson. This work was supported by NSF/ARPA/NASA under Stanford's Digital Libraries contract.

References

Almuallim, H.; Akiba, Y.; and Kaneda, S. 1996. An efficient algorithm for finding optimal gain-ratio multiple-split tests on hierarchical attibutes in decision tree learning. In *Proc. AAAI-96*, 703–708.

Chickering, D. M. 1995. Learning bayesian networks is NP-complete. In *Lecture Notes in Statistics.*

Cover, T. M., and Thomas, J. A. 1991. *Elements of Information Theory.* Wiley.

Fisher, D. H. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 2:139–172.

Friedman, N., and Goldszmidt, M. 1996a. Building classifiers using bayesian networks. In *Proc. AAAI-96*, 1277–1284.

Friedman, N., and Goldszmidt, M. 1996b. Learning bayesian networks with local structure. In *Proc. UAI-96*, 252–262.

Good, I. J. 1965. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods.* MIT Press.

Hersh, W. R.; Buckley, C.; Leone, T. J.; and Hickam, D. H. 1994. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proc. SIGIR-94*, 192–201.

Infoseek. 1995. Internet directory and query service. `http://www.infoseek.com/`.

Infoseek. 1996. Aptex categorizes more than 700,000 web sites for infoseek. `http://info.infoseek.com/doc/PressReleases/hnc.html`.

Koller, D., and Sahami, M. 1996. Toward optimal feature selection. In *Proc. ICML-96*, 284–292.

Pazzani, M. J. 1995. Searching for dependencies in bayesian classifiers. In *Proc. AI&STATS-95.*

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan-Kaufmann.

Sahami, M. 1996. Learning limited dependence bayesian classiers. In *Proc. KDD-96*, 335–338.

Schutze, H.; Hull, D.; and Pedersen, J. 1995. A comparison of document representations and classifiers for the routing problem. In *Proc. SIGIR-95*, 229–237.

Self, G. 1996. Personal Communication.

Singh, M., and Provan, G. M. 1996. Efficient learning of selective bayesian network classifiers. In *ICML-96*, 453–461.

van Rijsbergen, C. J. 1979. *Information Retrieval.* Butterworths.

Yahoo! 1995. On-line guide for the internet. `http://www.yahoo.com/`.