Title: Hierarchically classifying documents using very few words

Authors:

Daphne Koller	Mehran Sahami
Computer Science Department	Computer Science Department
Gates Building 1A	Gates Building 1A
Stanford University	Stanford University
Stanford, CA 94305-9010	Stanford, CA 94305-9010

Abstract: The proliferation of topic hierarchies for text documents has resulted in a need for tools that automatically classify new documents within such hierarchies. One can use existing classifiers by ignoring the hierarchical structure, treating the topics as separate classes. Unfortunately, in the context of text categorization, we are faced with a large number of classes and a huge number of relevant features needed to distinguish between them. Consequently, we are restricted to using only very simple classifiers, both because of computational cost and the tendency of complex models to overfit.

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 the computational and robustness difficulties described above.

Keywords: Information Retrieval, Hierarchical Classification, Feature Selection, Probabilistic Models

Email address of contact author: sahami@cs.stanford.edu

Phone number of contact author: (415) 725-8784

Multiple submission statement: This paper is currently not under review for another conference or journal, nor will it be submitted elsewhere during ML's review period.

Hierarchically classifying documents using very few words

Abstract. The proliferation of topic hierarchies for text documents has resulted in a need for tools that automatically classify new documents within such hierarchies. One can use existing classifiers by ignoring the hierarchical structure, treating the topics as separate classes. Unfortunately, in the context of text categorization, we are faced with a large number of classes and a huge number of relevant features needed to distinguish between them. Consequently, we are restricted to using only very simple classifiers, both because of computational cost and the tendency of complex models to overfit.

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 the computational and robustness difficulties described above.

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 classification hierarchies of this type have long been used in special-purpose collections of documents such as MEDLINE [HBLH94] or collections of patent documents [Sel96]. More recently, they have been used in several internet search engines, such as Yahoo [Yah95] or Infoseek [Inf95], to categorize the entire 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.[1] 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 [AAK96] which focused on hierarchically structured attributes rather than classes), most work in classification has ignored the problem of supervised in the presence of hierarchically structured classes. (There has been some work on unsupervised hierarchical clustering, e.g., [Fis87].)

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 tree. 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, based on the words that it does and does not contain, 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 large model will have many thousand parameters which need to be estimated, and thus can easily lead to overfitting of the training data.

Previous work [SHP95], has shown that feature selection can be a useful tool in dealing with this issue. We eliminate many of the words that appear in the corpus as being unindicative of the topic. The results of [KS96] show that one can obtain a significant increase in accuracy by reducing the number of words used for classification from 1600 to as few as 100. However, even for 100 features, the computational cost and the robustness still pose significant limitations, forcing us to use only very simple classifiers such as Naive Bayes [Goo65].

The flattened classifier loses the intuition that topics that are close to each other in the hierarchy have a lot more in common with each other, in general, than topics that are very far apart. Therefore, even when it is difficult to find

[1] Indeed, Infoseek has recently attempted to overcome this difficulty by using neural network technology to automatically categorize webpages [Inf96].

the precise topic of a document, it may be easy to decide whether it is about "agriculture" or about "computers".

Our approach, therefore, is to divide the classification task into a set of smaller classification tasks, each of which corresponds to some split 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.

The key insight is that 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. Thus, they achieve better accuracy even for very simple classifiers such as Naive Bayes. The use of smaller models also allows us to go beyond the simple classifiers, and search for a more complex classifier that more realistically models the data. For example, we can train a probabilistic classifier (such as TAN [FG96] or KDB [Sah96]) that takes into account the correlation between different features (e.g., the fact that Microsoft and Windows tend to co-occur). As we will see in our experimental results, the use of more expressive models allows us to obtain better accuracy. In contrast, we show that it is typically infeasible to learn such models in the presence of many features, and that, even when feasible, the overfitting problem results in reduced accuracy.

It is important to note that the key here is not merely the use of feature selection, but its integration with the hierarchical structure. We do not get the same performance if we simply choose a small feature set and use it for classification in the flattened class space. To understand this, 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 helped 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 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, any document only "sees" 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"). And even the features which it does see are divided so as to focus the attention of the classifier on the features relevant to the classification subtask at hand.

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 the disease that a patient has 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, respectively, 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 1280 to 10), and that the resulting classifier, in addition to being smaller and easier to train, also provides much 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 [KS96] and a range of classifiers of varying complexities and accuracies [Paz95, FG96, Sah96, SP96]. 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 [Pea88] 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 nondescendants 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

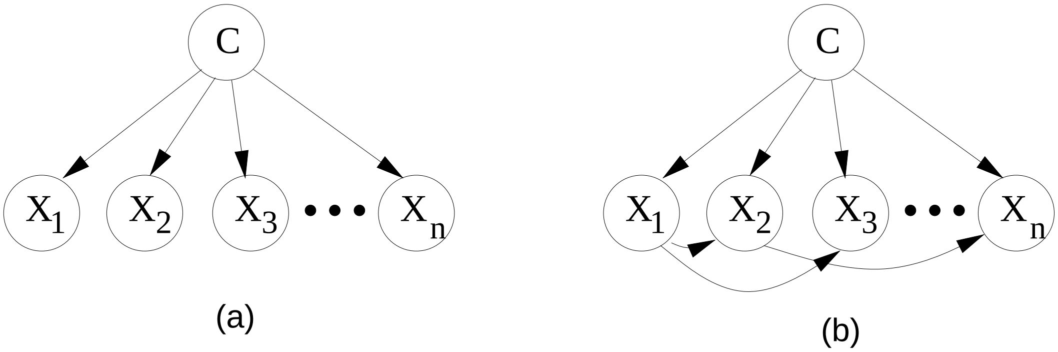

Figure 1: Bayesian networks corresponding to (a) a Naive Bayesian classifier; (b) A more complex Bayesian classifier allowing limited dependencies between the features.

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 (as in [SP96], empirical evidence [FG96] 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 [Goo65]. 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 mentioned in the introduction) 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 [Chi95]. Thus, any 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 [FG96] 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 [Sah96], 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, $I(X_i; X_j|C)$ [CT91].

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.)

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.

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. Thus, for text domains, the number of features can still be a bottleneck and feature selection becomes imperative, even if we did not consider 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 [KS96]. This method for feature selection employs Information Theoretic measures [CT91] to determine a subset of the original domain features that seem to best capture the class distibution in the data. Formally, the cross-entropy metric between two distributions μ and σ, defined as $D(\mu, \sigma) = \sum_{x \in \Omega} \mu(x) \log \frac{\mu(x)}{\sigma(x)}$, provides us with a formal notion of the "distance" between μ and σ. For each feature X_i, the algorithm determines the expected cross-entropy $\delta_i = P(X_i)D(P(C|\mathbf{X}), P(C|\mathbf{X_{-i}}))$ where $\mathbf{X_{-i}}$ is the set of all domain features except X_i. It then eliminates the features X_i for which δ_i is minimized. Thus, the feature eliminated least disrupts the original conditional class distribution. This process can be iterated to eliminate as many features as desired. To compute $P(C|\mathbf{X})$ the feature selection algorithm simply uses the Naive bayes model for speed and simplicity. Moreover, since this model assumes conditional independence of features, it is a fast process to update $P(C|\mathbf{X})$ to $P(C|\mathbf{X_{-i}})$ after the elimination of X_i. In this respect, the algorithm is very applicable to text domains with many features. Koller & Sahami have demonstrated this in experiments with text, and thus their method was selected for use here. Note that Koller & Sahami's method also provides a mechanism for eliminating features whose predictive information with respect to the class is subsumed by other features, but since this extension requires quadratic time with respect to the number of features, it was not employed here.

3 Experimental Methodology

In order to test our scheme for hierarchical classification, we first needed to obtain hierarchically classified text data. To this end, we made use of the Reuters-22173 dataset[2] and applied our own processing methods to construct two hierarchically classified document collections.[3]

The Reuters collection does not have a pre-determined hierarchical classification scheme, but each document can have multiple labels, so we identified labels which tended to subsume other labels. Two subsets of the Reuters collection, which we call Hier1 and Hier2, were then extracted, in which every document contained one (and only one) *major topic* and *minor topic* (or subtopic). The major topics were then all grouped together at the top level of the hierarchy. Since the entire Reuters collection deals with business related articles it would be fair to assume that the top level of each hierarchy could be labelled as "Current Business". The major and minor topics in the two datasets are described in Tables 1 and 2.

Next, each of these datasets was split 70%/30% into class stratified training

[2] 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.

[3] Ultimately, we hope to try out our approach on text from commercial Web directory hierarchies. We are currently negotiating with several such companies for research access to their data.

		Dataset size	
Major topics	minor topics	training	testing
Grain	Corn	115	50
	Wheat	156	67
Money Effects	Dollar	138	60
	Interest Rates	136	59
Crude Oil	Natural Gas	56	25
	Shipping	53	24
Total		654	285

Table 1: Description of the Hier1 dataset.

		Dataset size	
Major topics	Minor topics	training	testing
Acquisitions	C-bonds	13	6
	Earnings	11	6
Veg-Oil Bus.	Oilseed	21	9
	Palm Oil	25	12
Total		70	33

Table 2: Description of the Hier2 dataset.

and testing sets (as described in the figure). At this point, all document processing is done on the *training set only* so as not to create overly optimistic experimental results from having had any prior access to the testing data. We then apply 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 the training corpus. This is done so that we do not get unrealistic improvements in accuracy by simply eliminating features that are not even likely to appear in the test collection, or are so frequent that they will have no bearing on classification. Finally, each document is represented as a boolean vector, in which each feature denotes the presence or absence of a word that appeared in the training corpus and survived the initial Zipf's Law-based feature selection. These datasets were then used in our experiments, as detailed below.

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 major topics 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 major topic, the minor topics are used as class labels. For each major 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 each major 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 then 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 minor topic) 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 hope to address the issue of recovering from early errors in future work, but that would only serve to improve our results.

In the flat classification scheme, we simply treat every minor topic as a separate class. We then apply probabilisitic feature selection and induce one probabilistic model based on this reduced feature set.

4 Results

As an accuracy baseline, we ran the both the hierarchical and flat classification schemes on the datasets without employing any probabilistic feature selection. These results, as well as the original number of features in each dataset are

		Hierarchical			Flat		
Dataset	# Features	NB	KDB-1	KDB-2	NB	KDB-1	KDB-2
Hier1	1258	81.8%	81.4%	85.6%	83.2%	81.4%	83.5%
Hier2	283	78.8%	66.7%	66.7%	81.8%	57.6%	66.7%

Table 3: Baseline accuracy results for hierarchical and flat learning, employing only Zipf's law for feature selection.

given in Table 3. Here we observe two important phenomena. First, in the Hier1 dataset, the very large number of features used precludes the hierarchical scheme from performing better than the simple flat method. More importantly, in the Hier2 dataset, the large number of features and the small dataset size allows for the more expressive KDB algorithm to overfit the training data. These initial results provide an empirical motivation for the integration of feature selection.

Since it is our belief that a very small set of features suffices for accurately distinguishing between topics (and furthermore helps avoid overfitting), we employed a very aggressive feature selection policy. We reduced the feature set to 20 and then to 10 features. Recall, however, that, in the hierarchical case, a potentially very different set of 10 or 20 features is selected at each node in the hierachy. Therefore, the hierarchical method, as a whole, actually examines a much larger set of features. To compensate for this, we also tested the flat method on a feature set consisting of the 50 most relevant features. These results are given in Table 4.

Of initial interest is the substantial improvement in accuracy over the baselines results which did not include feature selection. In *every* run using feature selection, an improvement in accuracy was found, even though as many as 1248 features were eliminated for the Hier1 dataset! While this is an important result in itself, we are more interested in the difference between the hierarchical and flat classification methods.

Our results show that the hierarchical method clearly outperforms the flat classification method, when considering a direct comparison of the 10 and 20 feature runs. In 11 of the 12 runs, the hierarchical method produces a better classification accuracy than the corresponding flat method. Moreover, if reduction in relative error is compared, we find that the hierarchic method pro-

		Hierarchical			Flat		
Dataset	# Features	NB	KDB-1	KDB-2	NB	KDB-1	KDB-2
Hier1	10	92.6%	94.0%	93.3%	90.9%	89.8%	89.1%
Hier1	20	92.3%	93.0%	93.3%	91.6%	91.9%	92.3%
Hier1	50	—	—	—	92.3%	92.3%	93.7%
Hier2	10	87.9%	69.7%	90.9%	78.8%	72.7%	72.7%
Hier2	20	84.9%	84.9%	84.9%	81.8%	81.8%	78.8%
Hier2	50	—	—	—	78.8%	78.8%	72.7%

Table 4: Accuracy results for hierarchical and flat learning employing feature selection.

duces 8–41% fewer errors than the flat methods for Hier1 and somewhat more modest, but still substantial, relative gains for Hier2 (although the absolute accuracy gains are much clearer in this case).

The only exception is found when the hierarchical method is applied on the Hier2 dataset where 10 features are used in conjunction with the KDB-1 learning algorithm. A closer examination of this run reveals that, while no errors were made at the top classifier in the hierarchy, the classifier for the Acquisitions subtopic caused many of the classification errors. Given that this classifier was trained on only 24 instances, it is quite possible that a statistical anomaly in the data to which the algorithm was sensitive led to the induction of a poor classifier in this case. We see that even here, the flat method did not perform much better. In the Hier1 dataset, which has many more instances from which to glean statistical data, we dot not encounter such problems. In general, applications where automated hierarchical classification is desirable, such as Web directories, the volume of available data will help control for such anomalies.[4]

Finally, it is important to compare the results when the flat method is allowed to utilize 50 features, making it more comparable with the number of features that the hierarical method actually gets to see. Even in this case, the hierarchical method clearly outperforms the flat method in general, although by a smaller margin. Furthermore, when performing this comparison, we must

[4] Yahoo, for example, now claims to have close to 100,000 Web pages classified in their hierarchy.

Topic	10 most discriminating words
top level	bank, dollar, dealer, oil, agriculture, tonnes, grain, wheat, corn, usda
Grain	home, london, k, wheat, maize, corn, enhancement, winter, pl
Money Effects	dollar, pct, japan, tokyo, yen, money, repurchase, k, stg, system
Crude oil	price, production, ship, cubic, gas, natural, iran, barrel, attack, tanker

Table 5: The 10 most dicriminating words in the hierarchical method for the Hier1 dataset.

also keep in mind the computational costs. As we have seen, the complexity of algorithms for learning expressive classifiers grows rapidly with the number of features. Even when the grown is quadratic, as in the KDB and TAN algorithms, it is significantly more expensive to learn a single classifier over 50 features than to learn several classifiers over only 10 or 20 features each. Furthermore, if we wish to construct even more accurate models by using a real Bayesian network learning algorithm, this task may be achievable in the case of 10 features, but is clearly infeasible in the case of 50.

Our results also show that the feature selection stage does serve to focus the algorithm on the features relevant *to the local classification task*. Table 5 shows the set of 10 features (words) found to be most dicriminating at each level of the hierarchy learned for 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 longer indicative at the lower levels. Thus, for example while the term “agriculture” is useful for identifying documents in the Grain topic, it is 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 (which are generally measured in “pct”). 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”).

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 more accurate (but also 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. In particular, wse 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.

References

[AAK96] Hussein Almuallim, Yasuhiro Akiba, and Shigeo Kaneda. An efficient algorithm for finding optimal gain-ratio multiple-split tests on hierarchical attibutes in decision tree learning. In *Thirteenth National Conference on Artificial Intelligence*, pages 703–708, 1996.

[Chi95] D. M. Chickering. Learning bayesian networks id NP-complete. In *Lecture Notes in Statistics*, 1995.

[CT91] T.M̃. Cover and J.Ã. Thomas. *Elements of Information Theory.* Wiley, 1991.

[FG96] Nir Friedman and Moises Goldszmidt. Building classifiers using bayesian networks. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1277–1284, 1996.

[Fis87] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[Goo65] Irving John Good. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods.* M.I.T. Press, 1965.

[HBLH94] W. R. Hersh, C. Buckley, T. J. Leone, and D. H. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual ACM SIGIR Conference*, pages 192–201, 1994.

[Inf95] Infoseek.
Internet directory and query service. http://www.infoseek.com/, 1995.

[Inf96] Infoseek. Aptex categorizes more than 700,000 web sites for infoseek. http://info.infoseek.com/doc/PressReleases/hnc.html, 1996.

[KS96] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In Lorenza Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 284–292. Morgan Kaufmann Publishers, 1996.

[Paz95] M. J. Pazzani. Searching for dependencies in bayesian classifiers. In *Proceedings of the Fifth Int. Workshop on AI and Statistics*, 1995.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan-Kaufmann, 1988.

[Sah96] Mehran Sahami. Learning limited dependence bayesian classiers. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 335–338, 1996.

[Sel96] Glen Self. Personal Communication, 1996.

[SHP95] Hinrich Schutze, David Hull, and Jan Pedersen. A comparison of document representations and classifiers for the routing problem. In *Proceedings of the 18th Annual ACM SIGIR Conference*, pages 229–237, 1995.

[SP96] Moninder Singh and Gregory M. Provan. Efficient learning of selective bayesian network classifiers. In Lorenza Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 453–461. Morgan Kaufmann Publishers, 1996.

[Yah95] Yahoo! On-line guide for the internet. http://www.yahoo.com/, 1995.