Applying Hierarchical Classification Techniques Without A Hierarchy

Mark Chavira MCHAVIRA@CS.STANFORD.EDU
Dept of Computer Science, Gates Bldg 4A, 353 Serra Mall, Stanford, CA 94305-9040, USA

Dik Kin Wong DKWONG@STANFORD.EDU
Center for the Study of Language and Information, Stanford, CA 94305-4115, USA

Christopher Manning MANNING@CS.STANFORD.EDU
Depts of Computer Science and Linguistics, Gates Bldg 4A, 353 Serra Mall, Stanford, CA 94305-9040, USA

Abstract

The need to classify text documents within topic hierarchies has given rise to techniques that use the hierarchical structure to improve classification performance. We propose two methods, each utilizing information in a confusion matrix, which apply hierarchical concepts to problems where no a priori hierarchy exists. One method involves learning a hierarchy and then applying existing hierarchical techniques to it. The second method performs a second stage classification, where a second stage classifier attempts to correct common errors of the first stage classifier. Our experiments using the first technique show no improvement over our best flat classifier results, while those using the second technique produce minor improvements. Both sets of experiments suggest the possibility of more pronounced success in the future.

1. Introduction

Many useful sources of textual information now organize data into topic hierarchies. Examples include Yahoo! (Yahoo! 1995) and MEDLINE (Hersh et al. 1994). Because these databases often contain huge amounts of data, and because the amount of information typically grows over time, administrators and users can both benefit from automatic categorization. Because these databases often organize data into large numbers of topics, and because the distinctions among topics are not always great, the task of automatically categorizing such databases is often particularly difficult.

(McCallum et al., 1998) and (Mladenic and Grobelnik, 1997) have shown that when text documents are organized into topic hierarchies, classification techniques that take advantage of the hierarchical structure can provide better performance than standard flat classification techniques. In general, the approach has been to simplify the problem by dividing it into a number of simpler problems. Rather than training a single classifier that places a document into its final categorization, the hierarchical technique trains a classifier at each node in the hierarchy. Each classifier chooses from among the subtopics that are children of the topic to which the classifier corresponds. The final decision depends upon a series of simple classifications starting from the root, descending through interior nodes, and concluding at a leaf topic in the hierarchy.

The power of the hierarchical technique comes from the fact that each classifier has a simple job. Findings of (Koller and Sahami, 1996) suggest that there are two aspects to this simplicity. First, each classifier must consider only a limited number of topics. Second, the topics in this limited set all tend to be similar. When only some topics a classifier must distinguish are similar, that similarity makes classification more difficult. In contrast, when all topics are similar, the similarity can actually make the task easier. The reason is that most of the features among the topics will be similar and therefore of little use in classification. However, there will be a small number of telltale features that distinguish the topics. The limit on the number of features that are meaningful makes the classifier's job easier. It is important to note that similarity among all of the topics provides advantage, whereas similarity among some topics, as is the case when using a single flat-classifier, provides disadvantage.

We propose two methods of improving text categorization within data sets having flat topic sets. Both methods make use of information in a confusion matrix and of the two simplifications utilized by hierarchical classifiers. The first of these methods applies hierarchical techniques after learning a topic hierarchy. The second method performs a second stage classification, where a second stage classifier attempts to correct common errors of the first stage classifier.

2. Learning Hierarchies

If hierarchical classifiers outperform flat classifiers on data sets with topic hierarchies, then imposing and utilizing an appropriate hierarchy on a data set with flat topics might also improve classification performance. This hypothesis might be especially so if the flat topic set is large and if many of the topics look quite similar to each other. The task then is to determine how to generate an appropriate topic hierarchy from the training data and the flat topic set.

If we use standard hierarchical clustering techniques, as explained in (Jain, et al., 1999), with instances representing distinct topics, then the problem reduces to one of defining a similarity measure among the topics. We need to know which topics are "similar" to each other and should therefore be clustered together. One way to approach computing similarity among topics is to incorporate some notion of textual similarity. We suggest that a more efficient alternative, in terms of computational cost, is a measure of how often a flat-classifier confuses topics. If a flat-classifier confuses topics a, b, and c often, and it hardly confuses a, b, and c with other topics, then a classifier ought to at least be able to determine whether a given text document belongs to one of a, b, or c. In other words, a classifier in a classifier hierarchy ought to be able to accurately determine the cluster to which a given text document belongs. Then, taking advantage of reduction in problem complexity, a more focused, lower-level classifier should be able to distinguish among the similar topics in the cluster.

We can get information about confusions from the confusion matrix. In the discussion that follows, we orient our confusion matrices so that actual topics line the rows and predicted topics line the columns. Let #confusions (t_a, t_p) denote the entry in row t_a and column t_p of the confusion matrix. #confusions (t_a, t_p) therefore signifies the number of documents with actual topic t_a for which the classifier predicted topic t_p. We define the similarity between topics t_0 and t_1 to be as follows:

$$\text{similarity}(t_0, t_1) = \#\text{confusions}(t_0, t_1) + \#\text{confusions}(t_1, t_0)$$

We use the following procedure for generating our topic hierarchy:

1. Train a flat-classifier on training data. Call this classifier c_0.
2. Run the training data through c_0 to obtain a confusion matrix m_0.
3. Calculate the similarity between each pair of distinct topics using m_0 and the similarity measure defined above.
4. Form a topic hierarchy using hierarchical clustering techniques and the similarities defined in (3), where the instances are the categories.

With the newly generated topic hierarchy, we can apply hierarchical classification techniques. Specifically, we can:

1. Train a classifier at each node in the topic hierarchy.
2. Send each document being classified from the root, through interior nodes, and finally to a leaf, where its final categorization is determined.

3. Correcting Mistakes

If a flat classifier c_0 makes a number of mistakes and if we can determine how to correct a large number of those mistakes, then we can significantly improve classification performance. We believe that this technique might prove especially applicable if the flat topic set is large and if many of the topics look quite similar to each other. The task then is to (1) determine which mistakes c_0 makes often and (2) develop a means of correcting those mistakes.

We first need a way of determining which mistakes c_0 makes often. We define a mistake to be an incorrect classification. The kind of a mistake is defined by its (actual topic, predicted topic) pair. For example, a (t_a, t_p) mistake is a mistake in which c_0 predicts topic t_p but the actual topic is t_a ($t_p \neq t_a$). We can get the number of such mistakes from the confusion matrix. Let the confusion matrix be oriented as described in the previous section, so that actual topics make up the rows and predicted topics make up the columns. A classifier makes a (t_a, t_p) mistake often if the entry in row t_a and column t_p is large. More precisely, a classifier makes a (t_a, t_p) mistake often if its corresponding entry in the confusion matrix is among the k largest entries in column t_p, where k is a parameter which we can let vary across experiments. We discuss alternative ways of defining "often" toward the end of the paper.

Second, we need a method of correcting mistakes. If c_0 classifies a document as topic t_p, then the topic might actually be some different topic t_a. To help decide whether it is, we can make use of the simplifying techniques of the hierarchical classifier. That is, for each t_p, we can train a classifier $c_{1,p}$ solely on those topics which are often classified as t_p. These topics correspond to the rows of the k largest values in column t_p. Proceeding in this way makes $c_{1,p}$'s job significantly easier than c_0's job. That is, $c_{1,p}$ will have fewer topics to consider and all of the topics will be similar to each other. Once the training process is complete, we have a tree of classifiers of depth two. We can then extend the method to recursively produce a tree with greater depth and a

branching factor that gets smaller and smaller as we proceed further from the root.

The simplifications that the correcting plan makes for its second level classifiers are very similar to the simplifications the method of learning a hierarchy makes. Like the method of learning a hierarchy, the correcting method reduces the number of topics classifiers must distinguish. Also like the method of learning a hierarchy, the correcting plan uses a confusion matrix to group together for consideration topics that are similar and to remove topics from consideration that are not similar.

4. Experiments

We ran experiments on two data sets. Variants of these data sets have been used by various researchers, including (Weigend et al., 2000), (Koller and Sahami, 1996), and (McCallum et al., 1998).

The first data set is the 20 Newsgroups data set, collected by Ken Lang (Lang, 1995). This data set consists of about 20,000 articles, taken over a period of time from twenty Usenet Newsgroups. Each newsgroup contributed approximately 1,000 articles to the data set. We chose 100 random articles from each topic of this data set to use as a test set. We also stripped out headers from the articles. The data set then has twenty topics with 900 training documents and 100 test documents in each topic.

The second data set is the Reuters-21578 data set (Lewis 1997), with some modifications. The Reuters-21578 data set was taken from the Reuters newswire. It contains some documents that have multiple topics and some documents that have no topics. We stripped out these problematic documents and constructed a test set and training set according to the Lewis split, which is a split defined for the Reuters data set by its originators. The resulting data set has 59 topics. The number of documents in each topic varies largely. The number of training documents is 6552. The number of test documents is 2568.

We first ran a flat-level classifier on each data set to produce baseline results. We varied the number of features used, since our data sets were both very sensitive to this parameter.

We next ran experiments in which we learned a hierarchy for each of the data sets. For these experiments, we varied the number of clusters. Because the numbers of topics in our data sets were not extremely large, we made our tree have depth two.

We finally ran experiments in which we applied our method of correcting mistakes. In these experiments, we varied the number of features used by the c_0 classifier and the parameter k. Again, we restricted our tree to have depth two.

In all our experiments, we ran our tests using a variant of the Naive Bayes classifier. In general, Naive Bayes classifiers calculate the probability that a document d belongs to a topic t_i, given a set of topics T and a set of features F, according to the following formula:

$$P(t_i \mid d) = \frac{P(t_i) \times \prod_{f \in F} Score(f, t_i)}{\sum_{t \in T} P(t) \times \prod_{f \in F} Score(f, t)}$$

where $P(t_i)$ is the prior probability for topic t_i. Variants usually differ in how they compute Score(f,t). Our variant computes this value as follows:

$$Score(f, t) = 1 \quad \text{if f does not occur in d}$$

$$Score(f, t) = \frac{\#f}{\#F} \quad \text{otherwise}$$

where:

1. #f is the number of times f occurs in the training data in topic t_i.
2. $\#F = \sum_{f \in F} \#f$

Features are counts of words that occur in the documents. For the experiments using the Newsgroup data set, we removed features occurring fewer than four times, and we removed the most-common 100 features. For both data sets, we performed feature selection using mutual information.

5. Results

Figure 1 shows results for the Newsgroup experiments that learned a hierarchy. The baseline results shown are the best performance we could get from a single Naive Bayes classifier. The hierarchical results are the best results we could get by varying the number of clusters. The hierarchical method produces results comparable to the flat classifier. In general, performance tends to improve and approach the performance of the flat classifier as the number of clusters approaches the number of topics. We omit the results of the Reuters experiments that learned a hierarchy, because they are very similar to the Newsgroup results.

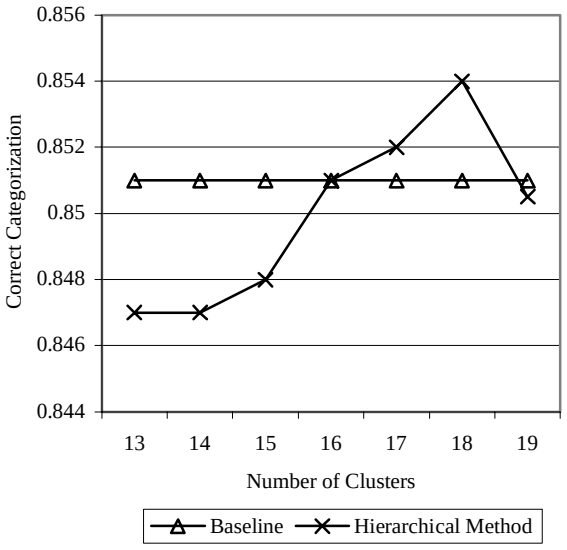

Figure 1. Newsgroup Learned Hierarchy Results.

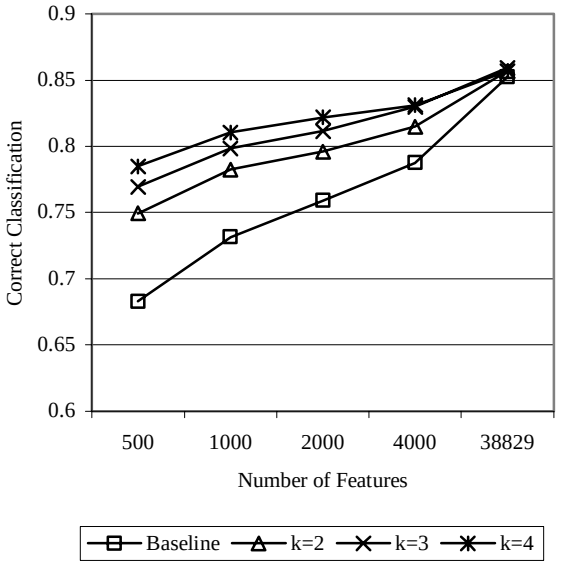

Figure 2. Newsgroup Correcting Mistakes Results.

Figure 2 shows results for the Newsgroup experiments that corrected errors. The baseline results show the best performance we could get from a single Naive Bayes classifier given a specific number of features. The other results show performance obtained using the correcting method with various values of k and various numbers of features. We obtained, in general, better results than the baseline. However, the more features we use, the closer all of the results become, up until they all become almost indistinguishable when we use all 38829 possible features.

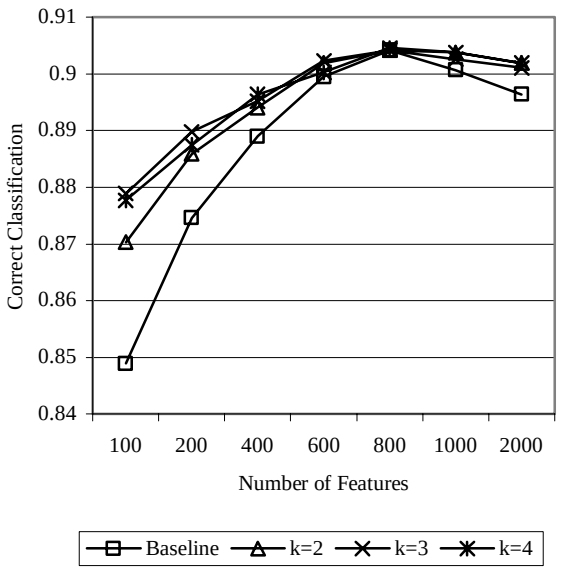

Figure 3. Reuters Correcting Mistakes Results

Figure 3 shows results for the Reuters experiments that corrected errors. It bears much in common with *Figure 2*. Again, we obtained, in general, better results than the baseline, and again, the best results using a single classifier are virtually equivalent to the best results using correcting classifiers. However, in the Reuters data set, as the number of features continue to increase, the various classifiers diverge again.

6. Discussion

The results of our learning a hierarchy experiments do appear significantly different from the results of a single flat classifier. However, the results of our correcting method suggest two interesting trends, worthy of further investigation.

First, we notice how the performances of the baseline and hierarchical techniques tend to begin far apart, get closer as the number of features increases, and then diverge again, once some limit is reached. Second, we notice that flat level classifiers tend to be more sensitive to tuning of parameters, while the hierarchical classifier tends to perform well across a larger assignment of parameters.

Our primary explanation for why improvements were not more pronounced is that we believe that the data sets we used are probably not the best for testing our methods: they either have too few categories or they have so few documents in some categories that they effectively have too few categories. Because they do not have a very large number of categories, a single classifier is able to do a very good job at the classification task. Improvement is more difficult. This state is evidenced by the fact that we were able to train single classifiers that achieved 85% and 90% on the Newsgroup and Reuters data sets,

respectively. We believe that our methods would work better on data sets that are sufficiently complicated to make a single classifier's results poor.

Our results indicate that, at least for modest-sized classifications and contrary to our expectations, our techniques produce only small benefit. However, we speculate that for larger hierarchies, results will be improved.

7. Further Work

We are currently working on several ideas to improve performance. Our primary idea is to test our theories on different data sets, for reasons described in the previous section. We would like to obtain a data set used previously for hierarchical work, flatten it, and then apply our techniques to see whether we can produce comparable or better results than researchers who trained hierarchical classifiers on the built-in topic hierarchy. Other ideas follow.

When learning a topic hierarchy, we would like to try other similarity measures. One idea is to sum the feature vectors for all instances in a class and to cluster the resulting vectors.

When applying our correction method, we use a very simple rule for determining how many topics an internal classifier should distinguish. Given a prediction t_p, we say that we should distinguish among the k topics that most commonly get classified as topic t_p. We have discovered that selecting different k values for different correcting classifiers can produce better results. We would like to pursue this idea further.

Also when applying our correction method, we would like to explore modifying our training process as follows. When we train an internal classifier, we choose our topics and then feed all training instances from those topics to the classifier. We have the idea that assigning more weight to training instances that actually reach the classifier would improve performance. Preliminary tests show small improvements. We would like to investigate further.

Acknolwedgements

This work is based in part on work supported by the National Science Foundation under Grant No. IIS-0085896.

References

Koller, D., and Sahami, M. (1996). Toward Optimal Feature Selection. In *Proc. ICML-96*, 284-292.

Lang, K. (1995). 20 Newsgroup data. http://www.ai.mit.edu/~jrennie/20_newsgroups/

Lewis, D. (1997). Reuters-21578 data set. http://www.research.att.com/~lewis/reuters21578.html

McCallum, A., Rosenfeld, R., Mitchell, T., and Ng, A. (1998). Improving Text Classification by Shrinkage in a Hierarchy of Classes. In *Proc. ICML-98*, 359-367.

Mladenic, D., and Grobelnik, M. (1997). Feature Selection for Classification Based on Text Hierarchy. In *Proc. ICML-97*, 170-178.

Mladenic, D. 1998. Turning Yahoo into an Automatic Web-Page Classifier. In *13th European Conference on AI*, pp. 473-474.

Sahami, M. (1996). Learning Limited Dependence Bayesian Classifiers. In *Proc. KDD-96*, 335-338.

Weigend, A., Wiener D., and Perdersen J. (2000). Exploiting Hierarchy in Text Categorization, *Information Retrieval Vol. 1, No. 2.*

Yang, Y. (1999). An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval* 1:69-90.

Jain, A.K, Murty, M.N., Flynn, P.J. (1999). Data Clustering: A Review. *ACM Computing Surveys, Vol. 31, No. 3, September 1999*.

On-line guide for the internet. http://www.yahoo.com.

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.