The Effect of Using Hierarchical Classifiers in Text Categorization

Stephen D'Alessio, Keitha Murray, Robert Schiaffino
Department of Computer Science
Iona College
New Rochelle, N.Y. 10801, USA
{sdalessio, kmurray, rschiaffino}@iona.edu

Aaron Kershenbaum
Department of Computer Science
Polytechnic University
Hawthorne, N.Y. 10532, USA
akershen@duke.poly.edu

Abstract

Given a set of categories, with or without a preexisting hierarchy among them, we consider the problem of assigning documents to one or more of these categories from the point of view of a hierarchy with more or less depth. We can choose to make use of none, part or all of the hierarchical structure to improve the categorization effectiveness and efficiency. It is possible to create additional hierarchy among the categories. We describe a procedure for generating a hierarchy of classifiers that models the hierarchy structure. We report on computational experience using this procedure. We show that judicious use of a hierarchy can significantly improve both the speed and effectiveness of the categorization process. Using the Reuters-21578 corpus, we obtain an improvement in running time of over a factor of three and a 5% improvement in F-measure.

1. Introduction and Background

The document categorization problem is one of assigning newly arriving documents to one or more preexisting categories. The focus of the research described below is the case where there is a hierarchy of preexisting categories. While in general, a lower level category may belong to more than one higher level category and the categories form a directed acyclic graph, in the problem we consider here the hierarchy is a tree. A document may belong to any number of low-level categories.

Until recently, most problems discussed dealt with categorization within a non-hierarchical set of categories. The Reuters-21578 corpus (available at http://www.research.att.com/~lewis) has been studied extensively. Yang (1997) compares 14 categorization algorithms applied to this Reuters corpus as a non-hierarchical categorization problem. Others treating the categories as a hierarchy (Chakrabarti, *et al.*, 1997; Koller & Sahami, 1997; Ng *et al.*, 1997; Yang 1996) have also studied this same corpus. Yang (1996, 1997) examines the OHSUMED corpus of medical abstracts. Still others examine the categories as a hierarchy for other corpora, namely the Yahoo Web hierarchy (McCallum *et al.*, 1998; Mladenic & Grobelnik, 1998).

Precision, recall, and F-measure are used by most authors as measures of the effectiveness of algorithms. Most simpler methods achieved values for these near 80% for the Reuters (Apte *et al.*, 1994; Cohen & Singer, 1996). More computationally expensive methods using the same corpus achieved results near 90% (Koller & Sahami, 1997) while methods that used hierarchy obtained small increases in precision and large increases in speed (Ng *et al.*, 1997).

We describe a hierarchical document categorization algorithm where the vocabulary and term weights are associated with categories at each level in the taxonomy and where the categorization process is iterated over levels in the hierarchy. A given term may be a discriminator at one level in the taxonomy receiving a large weight and then become a stopword at another level. The algorithm uses the hierarchy for feature

selection, reducing the number of terms associated with each category, resulting in a more efficient procedure.

Methods that incorporate additional feature selection have been studied (Apte *et al*., 1994; Chakrabarti *et al*., 1997; Deerwester *et al*., 1990; Koller & Sahami, 1996; Lewis, 1992; Ng *et al*., 1997; Yang & Pederson, 1997). The effectiveness of these methods varies. Most reduce the size of the feature set by one to two orders of magnitude without significantly reducing precision and recall. Some approaches assign weights to the features and then assign category ranks based on a sum of the weights of features present. Some weigh the features further by their frequency in the test documents. These methods are all known as linear classifiers and are computationally simplest and most efficient, but they sometimes lose accuracy because of the assumption they make that features appear independently in documents. More sophisticated categorization methods base the category ranks on groups of terms (Chakrabarti *et al*., 1997; Heckerman, 1996; Koller & Sahami, 1996; Sahami, 1996; Yang, 1997). The methods that approach the problem hierarchically compute probabilities and make the categorization decision one level in the taxonomy at a time.

We focus here on a categorization procedure that makes extensive use of the hierarchy. There are three strong motivations for taking the hierarchy into account. First, experience to date has demonstrated that both precision and recall decrease as the number of categories increases (Apte *et al*., 1994; Yang, 1996). One of the reasons for this is that as the scope of the corpus increases, terms become increasingly polysemous. This is particularly evident for acronyms, which are limited by the number of 3- and 4-letter combinations, and which are reused from one domain to another.

A second motivation for doing categorization within a hierarchical setting is it affords the ability to deal with very large problems. As the number of categories grows, the need for domain-specific vocabulary grows as well. Thus, we quickly reach the point where the index no longer fits in memory and we are trading accuracy against speed and software complexity. On the other hand, by treating the problem hierarchically, we can decompose it into several problems each involving a smaller number of categories and smaller domain-specific vocabularies and perhaps yield savings of several orders of magnitude.

An additional motivation emerges from this work. Introducing intermediate hierarchy levels through the aggregation of lower level categories, makes the decision at some levels become one of selecting a single category for each document. This can be done with high accuracy. While documents may belong to more than one category when the category set is considered as a whole, they are most often part of only one category when the problem is viewed hierarchically. This can occur in two ways. Most documents should be about a single major topic. As such, they would either be in a single category or else in several closely related subcategories. In the latter, they would be single category documents at all hierarchy levels except one, the one where the related subcategories are siblings. The other likely case is that the categories themselves are orthogonal and that the document is in a category in each dimension of the orthogonal categorization hierarchy. For example, in a corpus of news stories, one portion of the hierarchy may deal with "geographical areas" while another portion might deal with "subject areas". A document might then be "multi-category" in the sense that it can be categorized along each of these axes. Again the document would be a multi-category document only at the top level of the hierarchy where these multiple axes are considered. While there are exceptions to these general rules, for the most part considering the problem hierarchically reduces almost all the decisions to one-of-M decisions which can be made with higher precision than the binary decisions, especially when a large number of categories is involved. The experimental results presented in section 4 this paper support this idea.

In previous papers (D'Alessio *et al*., 1998*a*; D'Alessio *et al*., 1998*b*) we report on preliminary experiments using a corpus consisting of single category documents. Those experiments demonstrate that it is possible to obtain significant improvements in precision and recall by making use of the hierarchy. The principal focus then was on the effect of the hierarchy. Refining the hierarchy, in some cases, moving categories from one place within the hierarchy to another, improved the accuracy of the categorization. We also extended that investigation and focused on the effect of adjusting the category levels to further improve accuracy. We explored the situations where one approach worked better.

In this paper, we focus on modifying and supplementing the preexisting hierarchy. Our approach is a greedy algorithm that seeks to exploit the hierarchy one level at a time. It begins with a trivial hierarchy

with all categories under a single root and proceeds to introduce intermediate categories, guided by preexisting category aggregations or by the overlap of documents in multiple categories. The algorithm only introduces intermediate categories when there is a net gain in accuracy and speed, therefore guaranteeing performance at least as good as a simple non-hierarchical categorizer. For simplicity, the algorithm we present uses a simple linear classifier. It can, in fact, work with more sophisticated classifiers as well. Additionally the experiments described below deal with multi-category documents as well as single category documents.

2. Problem Definition

2.1. General Definition of Categories

We are given a set of categories where sets of categories may be further organized into supercategories. We are given a training corpus and, for each document, the categories to which it belongs. Documents can be single-category or multi-category.

2.2. Document Corpus & Taxonomy

We use the Reuters-21578 corpus, Distribution 1.0, which is comprised of 21578 documents, representing what remains of the original Reuters-22173 corpus after the elimination of 595 duplicates by Steve Lynch and David Lewis. The documents are "categorized" along five axes - topics, people, places, organizations, and exchanges. We consider only the categorization along the topics axis. Close to half of the documents (10,211) have no topic and as Yang (1996) and others suggest, we do not include these documents in either our training or test sets. Note, that unlike Lewis (acting for consistency with earlier studies), the documents that we consider no-category are those that have no categories listed between the topic tags in the Reuters-21578 corpus, leaving 11,367 documents with one or more topics. Most of these documents (9,495) are single-category documents while 1,872 are multi-category. The average number of topics per document is 1.26.

The Reuters-21578 collection uses 135 topics categories organized non-hierarchically. Additional information on the category sets available at Lewis' site describes an organization that has 5 additional categories that become supercategories of all but 3 of the original topics categories. Adding a root forms a 3-level hierarchy (see Figure 1). Documents are assigned only to leaf categories of the hierarchy.
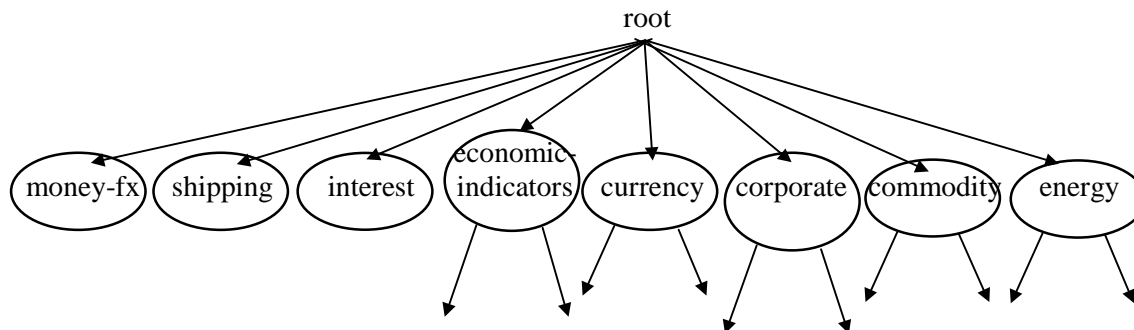

Figure 1: Reuters basic hierarchy

2.3. Performance Metrics

We measure the effectiveness of our algorithm using three standard measures. The first two are microaveraged precision (p) and recall (r), i.e., the ratio of correct decisions to the total number of decisions and the ratio of correct decisions to the total number of documents, respectively. We sometimes leave documents in non-leaf categories and then, in measuring precision and recall, count these as "no-category", reducing recall but not precision. The third, F-measure, is defined (van Rijsbergen, 1979) as:

$$F(p,r) = 2pr/(p+r)$$

3. Methods

3.1 Preparation of Data Sets

We begin by randomly dividing the corpus a training set (70%) and a validation set (30%) that is used as test data. The training set is then divided into two sets, a feature selection set and a parameter tuning set. If there are twenty or more training documents in a category, half the documents are placed in the feature selection set and half are placed in the parameter tuning set. Otherwise, all the documents are placed in both sets.

In building a categorizer we associate a set of positive features and a set of negative features with each category. We use the feature selection data to determine these features. Building the categorizer involves tuning a number of parameters using a procedure that will be described below. The parameter tuning data set is used for this purpose. Finally, the performance of the system is measured using the validation data set.

Before building the categorization system we preprocess all documents and convert them to a standard form. All characters are converted to lower case; all non-alphabetic characters except '-' are removed and all stop-words are removed. We construct a vocabulary from the training documents. The vocabulary consists of one and two word terms that occur ten or more times across the training set. We associate a list of features with each document in the training and the validation sets. A feature is added to a document's list if it appears in the document and also in the vocabulary, or if appears in the document two or more times even if it does not appear in the vocabulary. We associate a frequency with each feature in a document's feature list. Each appearance of a feature in the title of a document is counted as three occurrences. Each appearance of a feature in the body of a document is counted as one occurrence. The maximum frequency for a feature in a document is four.

3.2 The Basic Categorizer

The basic categorizer is used in situations where we are given a set of categories C_1, C_2, …, C_N with no hierarchical relation. We can think of this as a trivial hierarchy and we can model it by creating a root or parent category P and making each C_K a child of this root.

Therefore our basic categorization system consists of a parent category together with two or more child categories. Each category has two tunable parameters, a level number and a threshold. The root or parent category P has level $L_P = 1$ and threshold $T_p = 0$. Assume that the level of child $C_K = L_{CK}$, and that for all children C_K, $L_{CK} > L_P$. Assume the threshold for each child category C, T_C, is initialized to zero.

3.2.1 Feature Selection. Category features are selected using a variation of the ACTION algorithm described more completely in (Wong *et al*., 1996). For each document in the feature selection data set, and for each category that the document belongs to, add the document features and their frequencies to the category's positive feature list. The number of documents used in calculating the frequency of a feature in a category is capped at one hundred. Once all feature selection data has been processed we calculate an "effective frequency" for each positive category feature. The effective frequency of a positive feature f in a category C, denoted, EF_{fC}, is the same as the frequency of f in C if C is a child, and it is the sum of the frequency of f in C plus the frequency of f in all children of C if C is the parent. We then calculate the "significance" of each positive feature in a category. For all categories C, the significance of a positive feature f, denoted V_{fC}, is the effective frequency of the feature times the level number of the category. That is,

$$V_{fC} = EF_{fC} * L_C.$$

3.2.1.1 Selection of Negative Features. For each child category C_K, select the set of positive features that occur in the siblings of C_K but that do not occur in C_K. These features are the negative features for C_K. Note that the positive feature list of the parent category P contains all positive features that occur in any child category. Therefore the negative features for a child category can be found by comparing the feature list of the child with the feature list of the parent.

3.2.1.2 Selection of Positive Features. For each child C_K, and for each positive feature of C_K, compare the significance of the feature in the parent P with its significance in C_K. If the significance in C_K is less than or equal to its significance in P, the feature is removed from the positive feature list of C_K. Therefore a positive feature f remains in a category C if $V_{fC} > V_{fP}$ where P is the parent of C.

Each category C is assigned a maximum number of positive features Num_{PC} and maximum number of negative features Num_{NC}. For each category, the positive and negative features are ordered by significance and the most significant features are retained.

3.2.2 Feature Weights. Weights are assigned to the surviving features in each category. A weight W_{fc} is associated with each positive feature f in category C. W_{fc} is defined as:

$$W_{fc} = (\lambda + (1 - \lambda) * EF_{fc}/M_c)$$

where EF_{fc} is the effective frequency of f in C, M_c is the maximum frequency of any feature in C, and λ is a parameter currently set to 0.4. Similarly, a weight W_{fc} is associated with each negative feature f of category C. In this case, W_{fc} is defined as:

$$W_{fc} = -(\lambda + (1 - \lambda) * EF_{fp}/M_p)$$

where EF_{fp} is the effective frequency f with respect to its parent, and M_p is the maximum frequency of any feature in C's parent. As above λ is a parameter currently set to 0.4.

3.2.3 Types of Categorizers. The basic categorization system may be one of two types, binary or one-of-M. The type of categorization used by a system is determined by the training data for that system. When categorizing a document in a binary system the document may be assigned to more than one child category; in a one-of-M system the document may be assigned to at most one child category. With a binary categorizer the decision to place a document in a category C_i is made independently of the decision to place the document in any other category C_j. With a one-of-M categorizer the decisions are not independent. In our previous work (D'Alessio *et al.*, 1998*a*) with single category documents we found that one-of-M categorization was superior to binary categorization.

3.2.4 Document Categorization. When categorizing a document, D, a child category C is given document D by its parent and assigns a rank to D, R_{CD}. C compares this rank to its threshold T_C. If $R_{CD} > T_C$ the category returns a value of true to the parent; otherwise the child returns a value of false. The child also returns R_{CD} to the parent. R_{CD} is defined as:

$$R_{CD} = (\Sigma N_{fd} * W_{fc})/S_d$$

where the sum is taken over all positive and negative features associated with category C, N_{fd} is the number of times feature f appear in d, and S_d is the number of distinct features in d.

When binary categorization is used the parent assigns a document to all children that return a value of true. With one-of-M categorization the parent considers only children that return a value of true, and assigns the document to the highest ranking child. In both cases, if no child gives a document a ranking that exceeds its threshold it is unclassified.

3.2.5 Setting Level Numbers. The level number of a category is a tunable parameter. The relation between the level number of the parent category P and the level number of the child category C determines the positive features that are associated with the child. Recall that a positive feature f remains in a category C if $V_{fC} > V_{fP}$ where P is the parent of C. Since $V_{fC} = EF_{fC} * L_C$ a feature f remains in a child category C only if $EF_{fC}/EF_{fP} > L_P/L_C$. That is, ratio of a parent's level number and the child's level number is used as the lower bound on a feature's relative frequency in C. Changing the child's level number changes the minimum relative frequency and therefore changes the set of positive features associated with C.

We tune the level numbers by an iterative process. Recall that in the basic categorizer $L_P = 1$ and $T_P = 0$. For each child C_k we initialize $L_{CK} = 2 * L_P$, and $T_{CK} = 0$. Using these values and our feature selection

data set we execute our feature selection procedure. We then use the resulting system to categorize the parameter selection data set, and record the precision and recall of each category. We then evaluate the results and adjust the level numbers. As a general rule, if the recall of a category is low we raise the level number of the category; if the precision of the category is low we lower the level number of the category. By raising the level number of a child category we lower the relative frequency that a positive feature must have in order to remain in the child's feature list. This allows more features to remain with the child and increases the rank that the category gives to a document. Conversely, lowering the child's level number increases the relative frequency that a positive feature must have in order to remain in the child's feature list. Therefore more features are removed from the child, and generally the rank that the category gives to a document is reduced.

After the level numbers of the children have been adjusted, we repeat the feature selection and categorization procedures on the training data, and again examine the results and readjust the level numbers. This procedure is iterated until an acceptable level of recall and precision is achieved. At this stage we attempt to achieve high recall with acceptable precision. In the next stage we will tune the thresholds. Elevating the thresholds will increase the precision and decrease the recall.

If we are building a binary categorizer the adjustment of the level number of a category does not affect the behavior of any other category since independent decisions are made. In the case of a one-of-M categorizer this is not the case. Changing the level number of a category will change the positive features associated with that category and will affect the rank that a category gives to a document. With a one-of-M categorizer the document is given to the category that gives the document the highest ranking. Therefore changing the rank in one category may affect other categories.

3.2.6 Setting Thresholds. Each child category has an initial threshold of zero. Once the category's level number has been tuned, the threshold is tuned. The category ranks each document in the parameter tuning data set. Since this is training data, we know the categories the documents actually belong to. Beginning with a threshold value of 0 and incrementing in steps of 0.1, we determine the number of documents that would be correctly placed in the category with that threshold, and the number of documents that would incorrectly be placed in the category with that threshold. We associate a goodness measure with each threshold value by subtracting the number of incorrect documents from the number of correct documents. We select the threshold that corresponds to the maximum goodness value. If the same maximum goodness value occurs multiple times, we select the lowest threshold with that value. Once the thresholds have been set we have a complete categorizer.

3.3 The General Case

The basic categorization system described above is used both in the case of a trivial hierarchy and as the starting point for building more complex categorization systems. To build a hierarchical system of categorizers, we use a recursive process and construct a system top-down and breadth first.

Given a hierarchical set of categories, we start by considering a root level 1 with a threshold of 0, and a set of children. We determine the feature set, level number and threshold for each child as described above.

Now suppose there is a child C, with level L_C, threshold T_C, and a set of children. We use the procedures described above to create a categorization system for this sub-tree. Each child is initially assigned a level number $2*L_C$ and a threshold of zero. Features are selected and levels are determined as above but using only the training documents for categories of this sub-tree. Once the categorization system is built for this sub-tree, it is associated with the category C in the initial system. This results in a new set of leaves being added to the initial system. We repeat this procedure for each child that is the root of a sub-tree. When all such categories have been processed, we use the parameter tuning data to set thresholds for all the new leaves. We repeat this process as long as there are any categories that are roots of sub-trees.

3.4 Use of Hierarchical Structure

Now that we have a procedure for building a hierarchical categorization system, we can use it in a variety of ways. We might be given a corpus that has a hierarchical structure, such as the Reuters-21578 corpus with the topics hierarchy. We might choose to use all of the topics categories defined by Reuters and build a system of categorizers which models this hierarchy. We would supply a root and build a 3-level

hierarchical categorization system. We would have the option of building a binary or one-of-M categorizer for each sub-tree. That decision would be based on an examination of the training data. If the categories have largely disjoint training sets a one-of-M categorizer is indicated otherwise a binary categorizer is indicated.

We might have a corpus and a set of categories with no hierarchical structure. We may decide to create a hierarchy based on an examination of the training data. If the training documents can be partitioned into aggregated categories that are nearly disjoint, we can use these aggregated categories and build a one-of-M categorizer. Given a document we would first place it into one of these new categories and then decide which child it should be placed in. The second decision might also be binary or one-of-M.

A third possibility also exists. We may be given a corpus and a hierarchical set of categories. Examination of the training data may suggest that additional intermediate categories should be added to the hierarchy. Categorization of the training data might suggest that some of the given categories are not useful in the categorization process. We may examine alternative hierarchies, and create a hybrid consisting of some subset of the given non-leaf categories, and some categories discovered by an examination of the training data. A hierarchy can be built using a greedy procedure that adds or removes categories until no further improvements can be made.

4. Experiments and Results

In order to examine the effects of using hierarchical structures, we conducted a series of experiments with the Reuters-21578 corpus using our basic method described in Section 3 for constructing a system of categorizers. In each experiment, after building the categorizing system we used it to categorize the parameter tuning data, and the validation data. We do this to test for overfitting. We expect the performance of the categorization system to be better when we test with the parameter tuning data. However if the performance is significantly poorer when the validation data is categorized (which was not evidenced here) then overfitting is indicated. We report both results on both data sets below.

Our first experiment, E1, used no hierarchy (Figure 2). We defined a binary root category, and made all the Reuters categories children of this root. We constructed a categorizer for each child. Level numbers and thresholds were determined for each category as described above. The parameter tuning data was then categorized using the final categorizer. The F-measure on the training data was .828 and on the test data was .789.

We also repeated this experiment using a one-of-M root in place of the binary root. Using one-of-M categorization, a document could be assigned to at most one category. Since the average number of categories per document is 1.26, the maximum recall that could be achieved was approximately .80. However, since documents were assigned to fewer categories, we expected the precision to be higher. Again level numbers and thresholds were selected for each category. The parameter tuning data was then categorized and an F-measure of .736 was obtained. The same system was then used to categorize the test data and an F-measure of .770 was obtained. These results are reported below as experiment E1-M. Since experiment E1 produced better results it was used as the baseline.

In our second experiment, E2, (Figure 3) we examined the effect of imposing a simple hierarchy which emerged from an examination of the training data. In our training data there are 8,105 documents, of which 6,753 are documents belonging to a single category. The remaining 1,352 documents are multi-category documents and on average belong to 2.6 categories. We call each occurrence of a document in a category an instance of a document. With this definition there are 10,241 instances of documents in the training collection. Together, two categories, acquisitions and earnings, contain 4,463 single category documents. This is approximately half the documents, and about 40% of the document instances. These two categories have a very small intersection with each other and with the other categories (see Table 1). The remaining documents can be partitioned into two groups which we refer to as set-a and set-b. Set-a is an aggregation of the categories interest, money-fx, a group of categories that Reuters collectively calls currency, and another group of categories that Reuters collectively calls economic-indicators. Set-b is an aggregation of the categories shipping, a group of categories that Reuters collectively calls energy and another group that Reuters calls commodity. If we consider the training data as belonging to one or more of these four groups we find that there are 7,974 single category documents and only 131 multi-category documents, only one of which belongs to more than two categories. Table 1 shows the intersection of

these four categories. This suggests a simple hierarchy. These four categories can be made children of a root category. Set-a and set-b then become roots of sub-trees. Interest, money-fx, the currency categories and economic indicator categories become children of set-a. Shipping, the energy categories and the commodity categories become children of set-b.

We built a 3-level categorization system using this hierarchy. At the top level we used a one-of-M categorizer since the categories are virtually disjoint. In the sub-trees for set-a and set-b we used binary categorizers. Using this system we categorized both the parameter tuning data and the validation data. The results are shown in Table 2. There is an improvement in F-measure on both data sets. On the test data the F-measure increased from .789 in Experiment E1 to .821. It is encouraging that the recall is also better than in E1. There is some intersection between the categories at level 2. Making a one-of-M decision means that we cannot correctly categorize a document that belongs in more than one of the first level categories. There is a penalty for using a one-of-M categorizer. Our expectation was that there would be an improvement in precision. And in fact the improvement in precision was greater than the improvement in recall.

Experiment E2 differs from E1 in two respects. First, E2 uses a three level hierarchy (including the root); second, E2 makes a one-of-M decision at the first level. Either or both of these factors might account for the improved performance of the system in E2. We designed Experiment E3 to determine the effect of each factor. We built a categorization system that used the same hierarchy as in E2, but used a binary categorizer at the first level. We categorized both the parameter tuning and the validation data. The results of E3 are shown in Table 2. This system performed better than the system in E1, but not as well as the system in E2. This implies that making a one-of-M decision is responsible for the majority of the improvement seen is E2. Using a three level hierarchy provided some benefit, and allowed us to make the one-of-M decision.

We designed a set of experiments to discover if our categorization system could continue to achieve good performance as the hierarchy became more complex. We expanded our hierarchy by incorporating more of the Reuters topics hierarchy into our system.

The Reuters topics hierarchy considers acquisitions and earnings to be children of a corporate category. In our next experiment, Experiment E4, we modified the hierarchy from E2 by making corporate, set-a and set-b children of the root (Figure 4). Based on the results of the previous experiment we used a one-of-M categorizer for this first level. We now added a subtree with corporate as the parent and acquisitions and earnings as the children. Since acquisitions and earnings have a small intersection we built a one-of-M categorizer for this subtree. We continued to use binary categorizers for the set-a and set-b sub-trees. We hypothesized that using the corporate category at the first level would simplify the categorization at that level and result in more accurate results. However, in order to correctly categorize acquisitions and earnings documents it would now be necessary to make two decisions. Therefore there was the potential for a loss in precision and recall at the next level. As before we categorized both the parameter selection data and the validation data. The results are shown in Table 2. Both precision and recall are improved slightly from E2 and the F-measure increased from .821 to .828. Most significant is that adding a level to the categorizer did not adversely affect its performance.

Since adding the corporate category helped performance we decided to add the other Reuters topics categories. In our next experiment, Experiment E5, we expanded the set-a and the set-b categories (Figure 5). Set-a became the root of a sub-tree with interest, money-fx, currency and economic-indicators as children. Set-b became the root of a sub-tree with shipping, commodity and energy as children. Since there was no partition of the documents at this level we built binary categorizers for both set-a and set-b. The categories currency, economic-indicators, commodity and energy are themselves roots of sub-trees, and categorizers were constructed for each of them. We applied our techniques to each of these sub-trees. Within each sub-tree we found that there were disjoint categories and intersecting categories, so partitions did exist. Because of the small number of training documents in many categories we were unable to successfully build one-of-M categorizers for the partitions. We built binary categorizers for each of these sub-trees. The results are in Table 2.

The categorization system in E5 gave better precision but worse recall than the systems in E2 and E4. The F-measure declined slightly.

	acq	earn	set-a	set-b
acq		14	4	41
earn			0	17
set-a				57

Table 1. Number of Documents in the Intersection of Categories in Experiment 2

	Training Data				Test Data			Time per Doc (ms)
	Prec	Recall	F		Prec	Recall	F	
E1	83.8	81.9	.828		80.3	77.6	.789	235
E1-M	92.0	61.4	.736		89.5	67.7	.770	253
E2	88.5	82.4	.853		85.0	79.4	.821	83
E3	85.0	82.4	.837		81.4	78.3	.798	83
E4	88.4	83.0	.856		85.9	79.9	.828	86
E5	89.2	82.1	.855		86.4	79.0	.825	66

Table 2. Experimental Results

5. Validation

The experiments above were conducted on a single random partition of the Reuters corpus. To insure the validity of these results, experiments, E1 and E4 were replicated using a cross validation strategy. The original Reuters corpus was randomly divided into three sets. Experiments, E1 and E4 were then repeated six times. In each repetition a different permutation of the three sets was used for feature selection, parameter tuning and test data. In each case, the F-measure of E4 using the hierarchy was higher than the F-measure of E1 where no hierarchy was used.

The F-measures from the six repetitions of E1 were then compared with the six F-measures from the repetitions of E4 using a one way analysis of variance. The performance using the hierarchy was found to be significantly better than the performance without a hierarchy ($F = 8.06$, $p < .02$).

6. Summary and Conclusions

Starting with a trivial hierarchy and a simple binary categorizer we developed a series of increasingly complex hierarchies and categorizers. We used both one-of-M categorizers and binary categorizers where appropriate. In each case our categorization system was able to achieve precision and recall superior to that of the system for the trivial hierarchy. The F-measure improved by 5% when compared with applying the same categorizer to the trivial hierarchy. We demonstrated that an improvement in speed of over a factor of three, could be obtained by hierarchical categorization. From this we conclude that categorization systems can improve their performance by using hierarchies of categories. We also found that when a partition of the data exists, and there is sufficient training data, using one-of-M categorization can improve performance. The key factor in the improvement in speed of a one-of-M categorizer was that the average document is offered to a much smaller number of categories. In larger problems, with a larger number of categories, it is expected that this gain will increase proportionally.

While there was some improvement in accuracy due to reducing ambiguity in the feature set, as described in our previous work (D'Alessio *et al*., 1998*a*; D'Alessio *et al*., 1998*b*), our experiments demonstrated that the major improvement in accuracy was attributable to treating the problem at most intermediate nodes as a one-of-M problem. We found, however, that as we continued to introduce intermediate nodes further progress was not observed. We conjecture that this was because many of the categories had too few training documents resulting in wide variation in the category sizes. We conjecture that further gains may be obtainable with a larger corpus.
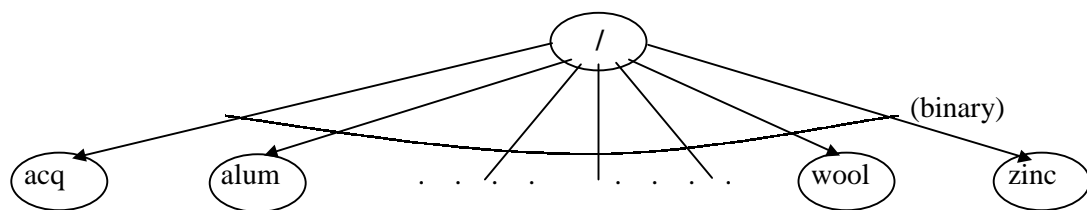
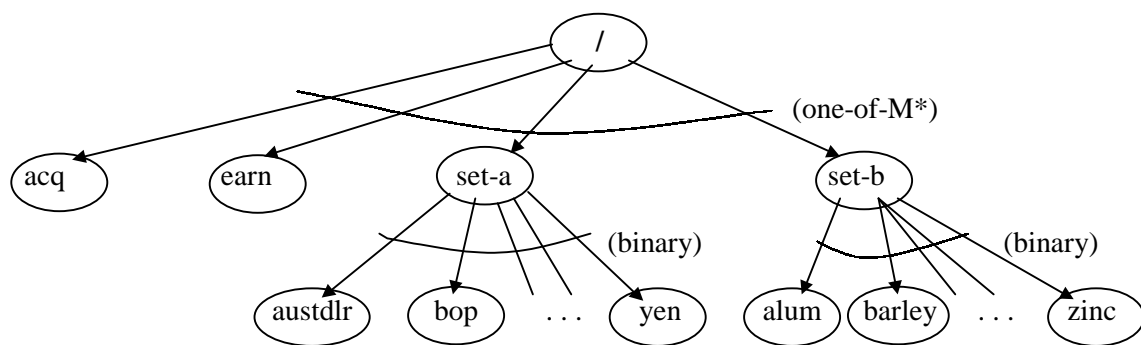Figure 2: Experiment E1 and Experiment E1-M

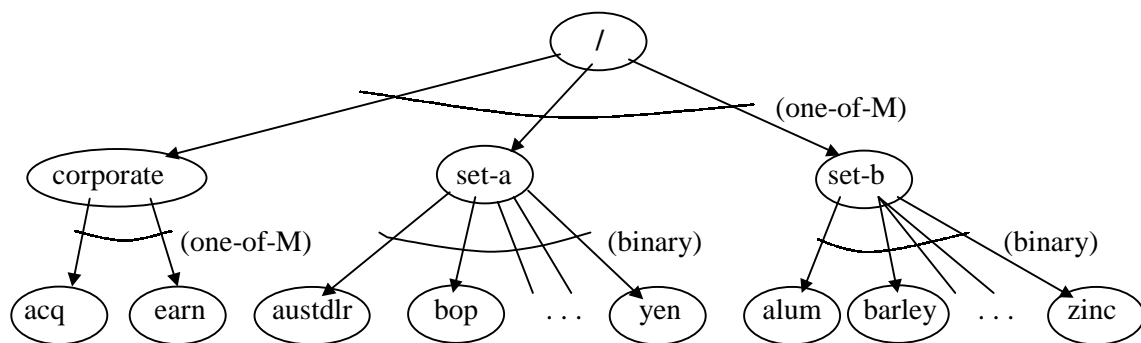Figure 3: Experiment E2 and *Experiment E3 (replacing one-of-M with binary)
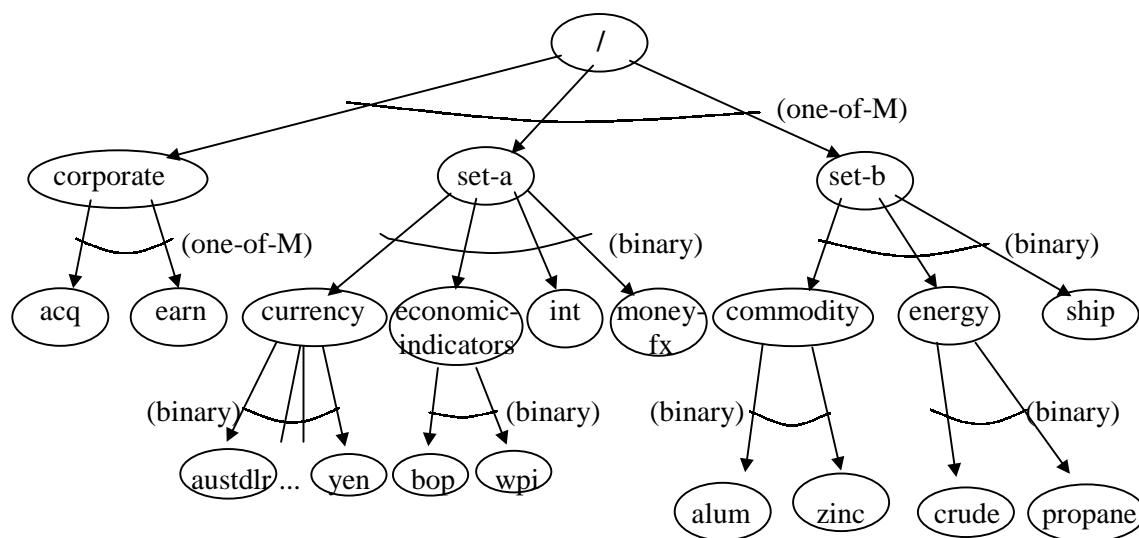

Figure 4: Experiment E4

Figure 5: Experiment E5

References

Apte C., Damerau F., Weiss S.M. (1994). Automated Learning of Decision Rules for Text Categorization. *ACM Transactions on Information Systems*, pp. 233-251.

Chakrabarti S., Dom B., Agarawal R., Raghavan P. (1997). Using Taxonomy, Discriminants and Signatures for Navigating in Text Databases. *Proceedings of the 23rd VLDB Conference*; Athens, Greece.

Cohen W.W. and Singer Y. (1996) Context-Sensitive Learning Methods for Text Categorization. *Proceedings of the 19th Annual ACM/SIGIR Conference*.

D'Alessio S., Kershenbaum A., Murray K., Schiaffino R. (1998). Category Levels in Hierarchical Text Categorization. *Proceedings of the Third Conference on Empirical Methods in Natural Language Processing (EMNLP-3)*.

D'Alessio S., Kershenbaum A., Murray K., Schiaffino R. (1998). The Effect of Topological Structure on Hierarchical Text Categorization. *Proceedings of the Sixth Workshop on Very large Corpora (COLING - ACL '98)*, pp. 66-75.

Deerwester S., Dumais S., Furnas G., Landauer T., and Harshman R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6), pp. 391-407.

Frakes W.B. and Baeza-Yates R. (1992). *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall.

Heckerman D. (1996). Bayesian Networks for Knowledge Discovery. *Advances in Knowledge Discovery and Data Mining*. Fayyad, Piatetsky-Shapiro, Smyth and Uthurusamy eds., MIT Press.

Koller D. and Sahami M. (1996). Towards Optimal Feature Selection. *International Conference on Machine Learning*, Volume 13, Morgan-Kauffman.

Koller D. and Sahami M. (1997). Hierarchically Classifying Documents using Very Few Words. *International Conference on Machine Learning*, pp.170-178, Volume 14, Morgan-Kauffman.

Lewis D. (1992). Text Representation for Intelligent Text Retrieval: A Classification-Oriented View. *Text-Based Intelligent Systems*, P.S. Jacobs, Lawrence-Erlbaum.

Lewis D. and Ringuette M. (1994). A Comparison of Two Learning Algorithms for text Categorization. *Third Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, pp. 81-93.

McCallum A., Rosenfeld R., Mitchell T., Ng A. (1998). Improving Text Classification by Shrinkage in a Hierarchy of Classes, *International Conference on Machine Learning*.

Mladenic D., Grobelnik M. (1998). Feature Selection for Classification Based on Text Hierarchy. *Working notes of Learning from Text and the Web, Conference on Automated Learning and Discovery CONALD-98*.

Ng H.-T., Goh W.-B. and Low K.-L. (1997). Feature Selection, Perception Learning and a Usability Case Study. *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Philadelphia, July 27-31, pp. 67-73.

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

van Rijsbergen C.J. (1979). *Information Retrieval*. Buttersworth, London, second edition.

Wong J.W.T., Wan W.K. and Young G. (1996). ACTION: Automatic Classification for Full-Text Documents. *SIGIR Forum* 30(1), pp. 11-25.

Yang Y. (1997). An Evaluation of Statistical Approaches to Text Categorization. *Technical Report CMU-CS-97-127, Computer Science Department, Carnegie Mellon University*.

Yang Y. (1996). An Evaluation of Statistical Approaches to MEDLINE Indexing. *Proceedings of the AMIA*, pp. 358-362.

Yang Y. and Pederson J.P. (1997). A Comparative Study of Feature Selection in Text Categorization, *International Conference on Machine Learning*, Volume 14, Morgan-Kauffman.