

The Effect of Topological Structure on Hierarchical Text Categorization

Stephen D'Alessio, Keitha Murray, Robert Schiaffino
Department of Computer and Information Science
Iona College
New Rochelle, N.Y. 10801, USA

sdalessio@iona.edu, kmurray@iona.edu, rschiaffino@iona.edu

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

ABSTRACT

The problem of assigning documents to categories in a hierarchically organized taxonomy and the effect of modifying the topology of the hierarchy are considered. Given a training corpus of documents already placed in one or more categories, vocabulary is extracted. The vocabulary, words that appear with high relative frequency within a given category, characterize each subject area by being associated with nodes in the hierarchy. Each node's vocabulary is filtered and its words assigned weights with respect to the specific category. Then, test documents are scanned for the presence of this vocabulary and categories are ranked with respect to the document based on the presence of terms from this vocabulary. Finally, documents are assigned to categories based on these rankings. Precision and recall are measured.

We present an algorithm for associating words with individual categories within the hierarchy and demonstrate that precision and recall can be significantly improved by solving the categorization problem taking the topology of the hierarchy into account. We also show that these results can be improved even further by intelligently selecting intermediate categories in the hierarchy. Solving the problem iteratively, moving downward from the root of the taxonomy to the leaf nodes, we improve precision from 82% to 89% and recall from 82% to 87% on the much-studied Reuters-21578 corpus with 135 categories organized in a three-level hierarchy of categories.

1. Introduction and Background

The proliferation of available online information attributable to the explosive use of the Internet has brought about the necessity for text retrieval systems that can assist the user in accessing this information in an effective, efficient and timely manner. Today's search engines have had difficulty keeping pace with the increasing amount of information that continuously needs to be indexed and searched. Categorization of the original text is a means by which the information can be arranged and organized to facilitate the retrieval task. Natural language processing systems can be used to query against these pre-specified categories yielding retrieval results more acceptable and beneficial to the user.

The document categorization problem is one of assigning newly arriving documents to categories within a given hierarchy of categories. In general, lower level categories may be part of more than one higher level category. Moreover, a document may belong to more than one low-level category. While the techniques described here can be applied to this more general problem, the experiments we have conducted, to date, have been carried out on a corpus where each document is a member of a single category and the categories form a tree rather than a more general directed acyclic graph. We limited the investigation to this more specific problem in order to focus the investigation on the effect of making use of the hierarchy, specifically on changes in the topology of the hierarchy.

Most computational experience discussed in the literature deals with hierarchies that are trees. Indeed, until recently, most problems discussed dealt with categorization within a simple (non-hierarchical) set of categories [6]. The Reuters-21578 corpus (available at David Lewis's home page: <http://www.research.att.com/~lewis>) has been studied extensively. Yang [21] compares 14 categorization algorithms applied to this Reuters corpus as a flat categorization problem on 135 categories. This same corpus has been more recently studied by others treating the categories as a hierarchy [2, 11, 15, 22]. Yang examines a portion of the OHSUMED [9] corpus of medical abstracts, a part of the National Library of Medicine corpus that has over 9 million abstracts organized into over 10,000 categories in a taxonomy (called MeSH) which is seven levels deep in some places.

We describe an algorithm for hierarchical document categorization where the vocabulary and term weights are associated with categories at each level in the taxonomy and where the categorization process itself is iterated over levels in the hierarchy. Thus 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 in the hierarchy. We also consider making modifications to the hierarchy itself as a means of increasing the accuracy and speed of the categorization process.

There are two 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 [1, 22]. 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 often limited by the number of 3- and 4-letter combinations, and which are reused from one domain to another.

The 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.

Feature selection, deciding which terms to actually include in the indexing and categorization process, is another aspect affected by size of the corpus. Some methods remove words with low frequencies both in order to reduce the number of features and because such words are often unreliable due to the low confidence in their distribution of occurrence across categories. Depending on the size of the corpus, this may still leave over 10,000 features, which renders even the simplest categorization methods too slow to be of use on very large corpora and renders the more complex ones entirely infeasible.

Methods that incorporate additional feature selection have been studied [1, 2, 5, 10, 13, 15, 24]. The effectiveness of these feature selection methods varies. Most reduce the size of the feature set by one to two orders of magnitude without significantly reducing precision and recall from what is obtained with larger feature sets. 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 the features appear independently in documents. More sophisticated categorization methods base the category ranks on groups of terms [2, 8, 11, 16, 21]. The methods that approach the problem hierarchically compute probabilities and make the categorization decision one level in the taxonomy at a time.

Precision and recall are used by most authors as a measure of the effectiveness of the algorithms. Most of the simpler methods achieved values for these near 80% for the Reuters corpus [1, 3]. More computationally expensive methods using the same corpus achieved results near 90% [11] while the methods that used hierarchy obtained small increases in precision and large increases in speed [15]. As the number of categories increased in a corpus (OSHUMED), precision and recall decline to 60% [22].

2. Problem Definition

2.1. Definition of categories

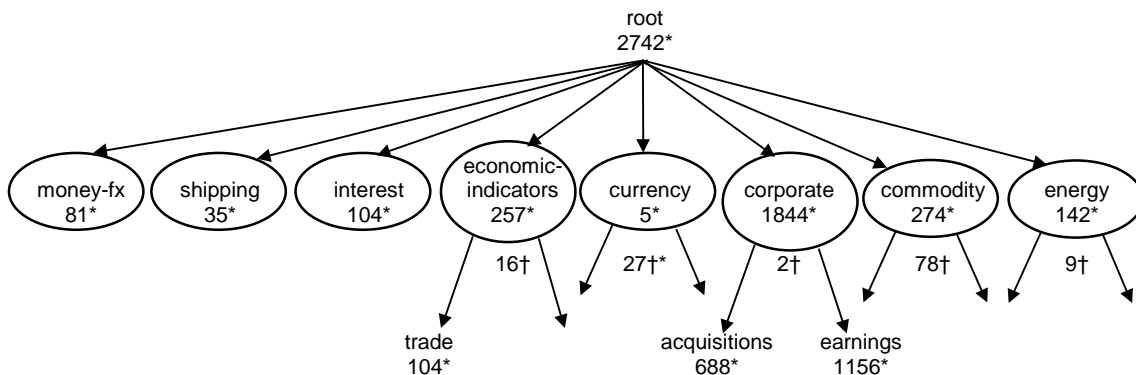
We are given a set of categories where sets of categories can be further organized into supercategories. We are given a training corpus and, for each document, the category to

which it belongs. Documents can, in general, be members of more than one category. In that case, it is possible to consider a binary categorization problem where a decision is made whether each document is or is not in each category. Here, we examine the M-ary categorization problem where we choose a single category for each document.

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 in 1996. The size of the corpus is 28,329,337 bytes, yielding an average document size of 1,313 bytes per document. 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 [22] 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' documents. This leaves 11,367 documents with one or more topics. Most of these documents (9,495) have only a single topic. The average number of topics per document is 1.26.

The Reuters collection uses a set of 135 categories organized as a flat taxonomy. Although the collection does not have a pre-defined hierarchical classification structure, additional information on the category sets available at Lewis's 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). Figure 1 includes counts by selected individual leaf categories and summarized by upper level supercategories. The number of categories per supercategory varies widely, from a minimum of 2 to a maximum of 78. The number of test documents per category also varies widely, from a minimum of 0 (for 76 such categories) to a maximum of 1,156 (earn). On the other hand, document size does not vary greatly across categories.



* number of test documents

† number of subcategories

Figure 1 Reuters basic hierarchy

In the same way that a wide variation in document size makes ranking documents with respect to a query in information retrieval difficult, it is difficult to accurately rank categories with respect to a document when the number of documents per category varies greatly across categories. Of course, we cannot control the number of documents actually in each category. We can reduce this variation to some extent by altering the hierarchy, at least temporarily, during the categorization process. Thus, for example, the hierarchy described in Figure 1 above group the “acq” and “earn” categories into a common supercategory "corporate". Each of these categories separately contains more documents than all of the other supercategories. Thus, we might improve the precision of the categorization process by "promoting" these categories to supercategories. This idea is explored in Section 4.

It might also help to temporarily move a category to a different part of the hierarchy when it shares important features with other categories there. In this case, by moving the categories under a common parent we can reliably get the document to that parent and then, using features that specifically separate these categories from one another, we can accurately complete the categorization. Moving categories is also explored in Section 4.

2.3. Performance Metrics

We measure the effectiveness of our algorithm by using the standard measures of microaveraged precision and recall; 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. We do, however, sometimes leave documents in non-leaf categories and then, in measuring precision and recall, count these as “no-category”, reducing recall but not precision.

3. Algorithm Description

3.1 Overview

We begin by creating training and test files using the 9,495 single-category documents from the Reuters-21578 corpus. While this led to somewhat higher precision and recall than would have been obtained by including multicategory documents, our 89% precision and 87% recall is also higher than the roughly 80% typically reported for categorization methods of comparable speed and complexity. Thus, our approach is comparable to those methods and serves as a reasonable baseline against which to study the effects of the hierarchy.

The corpus is divided randomly, using a 70%/30% split, into a training corpus of 6,753 training documents and 2,742 test documents. Documents in both the training and test corpora are then divided into words using the same procedure. Non-alphabetic characters (with the exception of "-") are removed and all characters are lowercased. Stopwords are

removed. The document is then parsed into “words”; i.e., character strings delimited by whitespace, and these words are then used as features.

Next, we count the number of times each feature appears in each document and, from that, we compute the total number of times each feature appears in training documents in each category. We retain only features appearing 2 or more times in a single training document or 10 or more times across the training corpus. All other features are discarded as being insufficiently reliable.

Next we use a variant of the ACTION Algorithm [20], described in detail in Section 3.2 below, to associate features with nodes in the taxonomy. This is one of the aspects that make our approach novel. This algorithm is particularly useful because it allows us to compare the frequency of a feature within a category with its frequency in sibling categories in the same subtree. This is more effective than just comparing the frequency within a category with global frequency as it focuses on the decision actually being made at that node in the hierarchy.

By eliminating most features from most categories, we gain several advantages. First, by limiting the appearance of a feature to a small number of categories (usually, just one) where it is an unambiguous discriminator, we improve the precision of the categorization process. Second, by working with a small number of features, we avoid optimization over a large number of features, and have a procedure with low computational complexity that can be applied to large problems with many categories. (Currently the number of features is set to 50). Our feature selection procedure most closely resembles rule induction [1] but it differs from that approach in that it considers the interactions among a larger number of features for a given amount of computational effort.

Weights are now assigned to the surviving features in each category. We associate a weight, W_{fc} , with each surviving feature, f , in category c . We define W_{fc} by:

$$W_{fc} = \left(\lambda + (1 - \lambda) \times N_{fc} / M_c \right) \quad (1)$$

where N_{fc} is the number of times f appears in c , M_c is the maximum frequency of any feature in c , and λ is a parameter (currently set to 0.4).

We also assign a negative weight to features associated with siblings (successors of the same parent node) of each category. A feature appearing in one or more siblings of c but not in c itself, is assigned a negative weight

$$W_{fc} = - \left(\lambda + (1 - \lambda) \times N_{fp} / M_p \right) \quad (2)$$

where p is the parent of c in the hierarchy. Thus N_{fp} is the number of times f appears in the parent of c , which is in turn the number of times f appears in all siblings of c since it

does not appear in c itself at all. M_p is the maximum frequency of any feature in c 's parent.

Finally, we filter the set of positive and negative words associated with each category, retaining, at most, 50 positive and 50 negative words with highest weights for each category, both leaf and interior.

We now have an index suitable for use in the category ranking process. The index contains features and a weight, W_{fc} , associated with each feature in each category. Note that W_{fc} is implicitly 0 for any feature not associated with a particular category.

Given a document, d , a rank can now be associated with each category with respect to d . Let F be the set of features, f , in D . The ranking of category c with respect to document d , R_{cd} , is then defined to be:

$$R_{cd} = \sum_f N_{fd} W_{fc} \quad (3)$$

where the sum is over all positive and negative features associated with c and N_{fd} is the number of times f appears in d . Note that, in practice, the sum is taken only over features that are in the intersection of the sets of features actually appearing in d and actually associated with c . Note that R_{cd} may be positive, negative or zero.

Test document d is now placed in a category. Starting at r , the root of the hierarchy, we compute R_{cd} for all c which are successors of r . If all R_{cd} are zero or negative, d is left at r . If any R_{cd} is positive, let c' be the category with the highest rank. If c' is a leaf node, d is placed in c' . If c' is an interior node, the contest is repeated at node c' . Thus, d is eventually placed either in a leaf category which wins a contest among its siblings or in an interior node none of whose children have a positive rank with respect to d . In this latter case, we may say that d is actually placed in the interior category, partially categorized or not categorized at all. Which of these we choose is dependent upon the application and on how much we value precision versus recall.

3.2 The ACTION Algorithm

The ACTION Algorithm was first described in [20] as a method of associating documents with categories within a hierarchy. Here, we use it to associate vocabulary with nodes in a hierarchy and associate documents with the nodes using the procedure described in Section 3.1 above. The original algorithm applied to problems with documents at interior and leaf nodes. Although our adaptations apply to the more general case also, we describe the algorithm with respect to that simpler case since the corpus we are using has documents only at leaf nodes.

The algorithm begins by counting N_{fc} , the number of times feature f appears in documents associated with category c in the training set, for all f and c . There is a level, L_c , associated with each category, c , in the hierarchy. By convention, the root is at level 1; its immediate successors are at level 2, etc.

We then define EF_{fc} , the effective frequency of a subtree rooted at node c with respect to feature f as

$$EF_{fc} = \sum_{j \in S_c} N_{fj} \quad (4)$$

Thus, EF_{fc} is the total number of occurrences of f in c and all subcategories, S_c of node c .

Finally, we define V_{fc} , the significance value of c with respect to f , as

$$V_{fc} = L_c \times EF_{fc} \quad (5)$$

Thus, a node gets credit, in proportion to its level, for occurrences of f in itself and in its successors. The farther down the tree a node is, the more credit it is given for its level, but the higher up the tree a node is, the larger the subtree rooted at c and the larger the credit it gets for effective frequency. A competition thus takes place between each node and its parent (immediate predecessor). For each feature, f , EF_{fc} is compared with, EF_{fp} , where p is the parent of c and if EF_{fc} is smaller then f is removed from node c . Thus a parent can remove a feature from a child but not vice versa. In the case of a tie, the parent keeps the feature. All this competition proceeds from the leaves upward towards the root.

The net effect of this is that if a feature occurs in only a single child of a given parent, then the child retains the feature (as does the parent), but if the feature occurs significantly in more than one child of the same parent, then only the parent retains the feature.

Several advantages accrue from all this. First, common features, including stopwords, will naturally rise to the root, where they will not participate in any rankings. Thus, this algorithm is a generalized version of removing stopwords. If a feature is prominent in several children of the same node, the parent will remove it from all of them. Ideally, words that are important for making fine distinctions among categories farther down in the category hierarchy, but are ambiguous at higher levels, will participate only in places where they can help.

Note that we never directly remove a feature from the parent even when the child retains it. The reason for this is that we may need the feature to get the document to the parent; if it doesn't reach the parent it can never reach the child. In the case where a feature strongly

represents only one category, there is no harm in the parent retaining it. In the cases where it is ambiguous at the level of the parent, the grandparent removes it from the parent (its child).

Thus, at the end of the algorithm when we filter the feature set for each category (leaf and non-leaf) retaining only the 50 most highly ranked positive and negative words, at non-leaf categories we also retain any words retained by their children.

4. Computational experience

There are a number of ways that the performance of a hierarchical categorization system can be tuned. Two alternatives that we are exploring are modifying the topology of the hierarchy and adjusting the weighting functions. This paper describes the experiments that we performed in order to understand the effects of modifying the topology. In another paper, we describe the effect of adjusting the level numbers (weights) of the categories within the hierarchy [4]. Our ultimate objective is to find a set of transformations that can be applied to a hierarchy as a part of the training process.

In the first experiment no hierarchy was used, that is, none of the 5 Reuters supercategories were used. We applied our feature selection algorithm and our categorization algorithm in the normal manner, however we assigned the root a level of 0. The effect of this is to prevent any features from being associated with the root. We refer to this organization as Flat-0. The remaining categories then keep their 50 most significant positive and negative features. The results for overall precision and recall, the number of unclassified documents (documents left at the root), and a selected example are reported in Table 1. Examining the results of this experiment shows that our algorithm does poorly in the case of several small categories. For example, there are only 4 petrochemical test documents, however our algorithm assigned 124 documents to the petrochemicals category of which only 1 was actually a petrochemicals document. Other small categories such as lumber, strategic metals, and money supply exhibit similar behavior. An examination of these categories shows that in each case they share a few key features with a larger category. When these features appear in a test document they are given disproportionate weight in the smaller categories. Of the incorrect documents assigned to petrochemicals, nearly all (118) were either acquisitions or earning documents. The vocabulary associated with the petrochemicals category in Flat-0 includes words such as “mln” and “dlrs” that are also earnings and acquisitions words. The formula used to assign weights to words found in test documents uses a normalization factor to account for the differences in the sizes of the categories. In this case the net effect is to bias the decision towards petrochemicals whenever these words appear in a test document.

One advantage of using a hierarchy is that it should provide a mechanism for moving features to positions where they aid in categorization and remove features from positions where they are ambiguous. We tested this hypothesis by introducing a simple hierarchical organization. We changed the level of the root node from 0 to 1, and gave

the subcategories of the root a level of 2. We refer to this organization as Flat-1. Again, each category kept its 50 most significant positive and negative features and the categorization algorithm was applied to the same test data as above. The comparison between Flat-0 and Flat-1, for this case, is also given in Table 1. Note the significant improvement in precision and recall. Examination of the vocabulary associated with the petrochemicals category in Flat-1 no longer includes "mln" and "dlrs" as the ACTION algorithm has removed them preventing this small category from stealing documents from larger categories with some similar features. Additionally, the time required for the categorization was reduced by a factor of one third. This experiment demonstrated the beneficial effect of using the ACTION algorithm with the hierarchy by allowing us to efficiently compare the relative frequency of features within a category and outside a category. The ambiguous words that were previously associated with petrochemicals were either moved to the root where they became stop words, or were moved to other categories.

	Root level	Overall Prec/Rec	Unclass Docs	Petrochem Correct	Petrochem Incorrect
Flat-0	0	82.85/82.79	2	1	124
Flat-1	1	89.36/85.74	111	0	0

Table 1: Comparison Between Flat-0 and Flat-1 Topologies

We then conducted a number of experiments to explore how modifying the topology of the hierarchy affects the categorization. As a baseline, we used the hierarchy of topics supplied with the Reuters corpus (see Figure 1) referred to as the Basic hierarchy. This organization is significantly different from Flat-1 in that it is a three-level hierarchy with 5 supercategories. We applied our feature selection and categorization algorithms using the same test data as above. The results for overall precision and recall, the precisions and recalls associated with the acquisitions and earnings categories themselves, and document placement counts are reported in Table 2 below. The time required for the categorization for the Basic hierarchy was approximately one half the time required for the Flat-1 case. An examination of the results shows that this hierarchy also avoids the small category problem experienced in Flat-0. However the overall performance was not as good as in Flat-1. We identified and analyzed situations where the use of the deeper hierarchy caused problems and attempted to study the problems by modifying the hierarchy.

An error analysis identified the first problem as occurring when sibling leaf categories steal documents from each other. An example is the case of the earnings and acquisitions categories. In the Basic hierarchy both earnings and acquisitions are subcategories of the corporate category while in Flat-1 both are subcategories of the root. A comparison of the precision and recall for acquisitions and earnings using Flat-1 versus Basic shows that

acquisitions' recall drops from 92% to 77% with the other values remaining somewhat comparable. In this case the deeper hierarchy impedes performance. An examination of the dispersion matrix (Table 2) for the Basic hierarchy shows that 91 acquisition documents are classified as earnings documents and 15 earning documents are classified as acquisitions documents with another 19 acquisition documents being left at the corporate node. This indicates that most of the earnings and acquisitions documents are being correctly classified as corporate documents, however, in many cases there is insufficient information to make the distinction between earning and acquisitions. We hypothesize that in this case, our vocabulary selection algorithm has removed too many terms from earnings and acquisitions and given them to corporate. Removing the corporate category from the hierarchy would allow earnings and acquisitions to become subcategories of the root and retain more of their significant features.

	Root	Corp	Acq	Earn	Interest	Trade	Eci*	Other
Root	0	0	0	0	0	0	0	0
Corp	0	0	0	0	0	0	0	0
Acq	25	19	529	91	0	8	1	15
Earn	2	1	15	1121	0	4	9	4
Interest	0	3	4	2	24	24	44	3
Trade	0	0	1	2	1	94	0	6
Eci*	2	1	0	7	0	21	122	9
Other	1	4	30	15	2	44	18	388

Eci* = Economic Indicators and all subcategories except Trade

Table 2: Dispersion Table for Basic Hierarchy

The columns list the categories where documents were placed by the algorithm; the rows list the categories the documents were actually in.

We tested this hypothesis by constructing a new hierarchy, Variation-1, by removing corporate from the Basic hierarchy. Table 3 summarizes the comparison between these two topologies. The table illustrates that in the case where acquisitions and earnings are both children of the root (Var-1) there is less stealing of documents occurring between these two siblings resulting in an overall improvement over the Basic hierarchy case.

	Overall Prec/Rec	Acq Prec/Rec	Earn Prec/Rec	Acq docs left at Corp	Earn as Acq	Acq as Earn
Basic	85.71/82.06	91/77	91/97	19	15	91
Var-1	87.55/84.61	92/90	97/94	-	38	16

Table 3: Comparison Between Basic and Var-1 Topologies

Basic - Acq and Earn are children of Corp and grandchildren of Root

Var-1 - Acq and Earn are children of the Root (no Corp node)

A second problem we identified is that in some cases the vocabulary selection process removes too many features from a leaf category with the result that it becomes difficult to properly categorize documents belonging to that category. An example of this can be seen with the category interest. As shown in the dispersion matrix for the Basic hierarchy above, there are 104 test documents belonging to the interest category, however only 24 interest documents are correctly classified. In this case interest is a subcategory of the root and most of its incorrectly classified test documents (68) are classified in the economic indicators subtree. Here we have a slightly different problem. We do not have sibling leaves stealing documents from each other. Only one economic indicators document, a trade document, is placed in interest. We have a leaf category competing directly with a larger, similar subtree. As a result many of its documents are placed in the subtree. We hypothesize that in this case the leaf category should be moved into the subtree. This would allow the smaller category to compete for the documents that are assigned to the subtree.

We tested this hypothesis by constructing a new hierarchy, Variation-2, by making interest a subcategory of economic indicators. Table 4 summarizes the comparison of the overall precision and recall, and selected document placement counts for the Basic hierarchy, Var-2, and a third hierarchy, called Var-3, that is a variation combining variations one and two. Again, we see an improvement in overall precision and recall but this time it was a result of making a category that was weak and losing its documents stronger by moving it to a position where it could directly compete for features and thus documents.

	Overall Prec/Rec	Int docs as Int	Non-Int docs as Int	Int docs placed incorrectly in eci subtree
Basic	85.71/82.06	24	3	68
Var-2	86.46/82.93	74	28	26
Var-3	88.72/85.78	76	30	27

Table 4: Comparison Among Basic, Var-2 and Var-3 Topologies

Basic - Interest is child of Root and sibling of Economic-Indicators

Var-2 - Interest is child of Economic-Indicators

Var-3 - Var-1 and Var-2 together

A third type of problem was also identified. At times a leaf category will have poor precision because it is assigned many documents not belonging to the category. In some cases this occurs because documents were incorrectly classified at a higher node in the hierarchy. These documents are then examined along the wrong path and are placed in an incorrect leaf. An example of this occurs in the category trade, which is the largest subcategory of economic indicators in the Basic hierarchy. The dispersion matrix shows that there are 104 trade test documents, 94 of which are correctly classified; 101 other documents are incorrectly classified as trade documents. This is not a case of a category

stealing documents from its sibling categories, rather documents belonging to a variety of non-economic indicator categories are incorrectly classified as economic indicators documents. When we have to decide which subcategory of economic indicators to place the documents into, trade being the largest subcategory attracts the majority of the documents. We hypothesize that we can correct this problem by moving trade and making it a subcategory of the root. This has two effects. First, it weakens economic indicators by removing one of its largest categories. Second, it weakens trade because it lowers its level number and therefore reduces the significance of its features. This is exactly the reverse of the actions that we took with interest, a category that was too weak to attract the documents it needed.

To test our hypothesis we constructed a new hierarchy, Variation-4, by making trade a subcategory of the root. We also incorporated our other variations, so that earnings and acquisitions are also subcategories of the root and interest is a subcategory of economic indicators. Table 5 reports the comparison of the Basic and Var-4 hierarchies. The overall precision and recall improve again, this time, by taking a category that is stealing because it was too strong and moving it to a position where it had to compete with equally strong siblings.

	Overall Prec/Rec	Trade docs as Trade	Non-Trade docs as Trade
Basic	85.71/82.06	94	101
Var-4	89.49/86.91	87	24

Table 5: Comparison Between Basic and Var-4 Topologies
 Basic - Trade is child of Economic-Indicators
 Var-4 - Trade is child of Root

Table 6 is a summary of the results for the all the hierarchies discussed above.

Hierarchy	Flat-0	Flat-1	Basic	Var-1	Var-2	Var-3	Var-4
Precision	82.85	89.36	85.71	87.55	86.46	88.72	89.49
Recall	82.79	85.74	82.06	84.61	82.93	85.78	86.91

Table 6: Precision (%) and Recall (%) for Seven Taxonomies

5. Summary

We have demonstrated that using a hierarchy can have a positive impact on the categorization task. Precision and recall are increased and the processing time is substantially reduced. In addition we have shown that the topology of the hierarchy can be modified to produce improvements in precision and recall. Our ultimate goal is to identify a set of transformations, category level settings, and the conditions under which

each should be applied so that we can automatically train the hierarchy. This would allow us to begin with a minimal hierarchy such as Flat-1, and, using training data, automatically evolve an optimal hierarchy. We are continuing to do research in this area.

An obvious danger when using a hierarchy is that placing a document into its correct category involves multiple decision points. If an error is made at an upper level in the hierarchy, the document will be placed incorrectly. Therefore it is critical that these early decisions be extremely accurate. Our experiments demonstrate that it is possible to achieve this accuracy. In the case of Flat-1 only one decision point is used and 2351 of 2742 (85.7%) test documents are placed in correct categories. In the case of Variation-4 if we look at only the first level, 2467 of the 2742 (89.7%) test documents are placed into the correct subcategory. In addition in Flat-1, the root is unable to make any decision for 111 (4%) documents while in Variation-4 there are only 23 (0.8%) such documents. On a supercategory basis, the root performed better for some than others. For commodities, it had precision and recall around 82%. For energy, it had about 93% precision and recall. Likewise, the performance of the interior nodes in the hierarchy varied. Economic indicators had a 88% precision and a 76% recall, while commodities had a 96% precision and a 93% recall. Thus we see that there is room for further improvement via moving categories from one part of the hierarchy to another and this investigation is the focus of our current research.

References

- [1] C. Apte, F. Damerau, S. M. Weiss. Automated Learning of Decision Rules for Text Categorization. *ACM Transactions on Information Systems*, 233-251, 1994.
- [2] S. Chakrabarti, B. Dom, R. Agarawal, P. Raghavan. Using Taxonomy, Discriminants and Signatures for Navigating in Text Databases. *Proceedings of the 23rd VLDB Conference*; Athens, Greece, 1997.
- [3] W.W. Cohen and Y. Singer. Context-Sensitive Learning Methods for Text Categorization. *Proceedings of the 19th Annual ACM/SIGIR Conference*, 1996.
- [4] S. D'Alessio, A. Kershenbaum, K. Murray, R. Schiaffino. Category Levels in Hierarchical Text Categorization. *Proceedings of the Third Conference on Empirical Methods in Natural Language Processing (EMNLP-3)*, June 1998 .
- [5] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6), pp. 391-407, 1990.
- [6] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, 1992.
- [7] M. Hearst, J. Pederson, P. Pirolli, H. Schutze, G. Grefenstette, and D. Hull. Xerox TREC4 Site Report in *Proceedings of the Fourth Text Retrieval Conference*, TREC-4, 1996.
- [8] D. Heckerman. Bayesian Networks for Knowledge Discovery. *Advances in Knowledge Discovery and Data Mining*. Fayyad, Piatetsky-Shapiro, Smyth and Uthurusamy eds., MIT Press, 1996.
- [9] W. Hersh, C. Buckley, T. Leone and D. Hickman. OHSUMED: An Interactive Retrieval Evaluation and a New Large Text Collection for Research. *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Philadelphia, 1994.
- [10] D. Koller and M. Sahami. Towards Optimal Feature Selection. *International Conference on Machine Learning*, Volume 13, Morgan-Kauffman, 1996.
- [11] D. Koller and M. Sahami. Hierarchically Classifying Documents using Very Few Words. *International Conference on Machine Learning*, Volume 14, Morgan-Kauffman, 1997.
- [12] L. Larkey and W.B. Croft. Combining Classifiers in Text Categorization. *Proceedings of the 19th Annual ACM/SIGIR Conference*, 1996.

- [13] D. Lewis. Text Representation for Intelligent Text Retrieval: A Classification-Oriented View. *Text-Based Intelligent Systems*, P.S. Jacobs, Lawrence-Erlbaum, 1992.
- [14] D. Lewis and M. Ringuette. A Comparison of Two Learning Algorithms for text Categorization. *Third Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, 1994, pp. 81-93.
- [15] H.-T. Ng, W.-B. Goh and K.-L. Low. 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, 1997, pp. 67-73
- [16] M. Sahami. Learning Limited Dependence Bayesian Classifiers. *Proc. KDD-96*, pp.335-338.
- [17] G. Salton. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*, Addison-Wesley, 1989.
- [18] C.J. van Rijsbergen. *Information Retrieval*. Butterworth, London, second edition, 1979.
- [19] I.H. Witten, A. Moffat and T. Bell. *Managing Gigabytes*. Van Nostrand Reinhold, 1994.
- [20] J.W.T. Wong, W.K. Wan and G. Young. ACTION: Automatic Classification for Full-Text Documents. *SIGIR Forum* 30(1), pp. 11-25, Spring 1996.
- [21] Y. Yang. An Evaluation of Statistical Approaches to Text Categorization. *Technical Report CMU-CS-97-127, Computer Science Department, Carnegie Mellon University*, 1997.
- [22] Y. Yang. An Evaluation of Statistical Approaches to MEDLINE Indexing. *Proceedings of the AMIA* , pp. 358-362, 1996.
- [23] Y. Yang and C.G. Chute. A Linear Least Squares Fit Mapping Method for Information Retrieval from Natural Language Texts. *Proceedings of COLING '92*, pp. 447-453, 1992.
- [24] Y. Yang and J.P. Pederson. Feature Selection in Statistical Learning of Text Categorization, *International Conference on Machine Learning*, Volume 14, Morgan-Kaufman, 1997.
- [25] UMLS Knowledge Sources 8th Edition; National Library of Medicine, January, 1997.