

Category Levels in 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

We consider the problem of assigning level numbers (weights) to hierarchically organized categories during the process of text categorization. These levels control the ability of the categories to attract documents during the categorization process. The levels are adjusted in order to obtain a balance between recall and precision for each category. If a category's recall exceeds its precision, the category is too strong and its level is reduced. Conversely, a category's level is increased to strengthen it if its precision exceeds its recall.

The categorization algorithm used is a supervised learning procedure that uses a linear classifier based on the category levels. We are given a set of categories, organized hierarchically. We are also given a training corpus of documents already placed in one or more categories. From these, we extract vocabulary, words that appear with high frequency within a given category, characterizing each subject area. Each node's vocabulary is filtered and its words assigned weights with respect to the specific category. Then, test documents are scanned and categories ranked based on the presence of vocabulary terms. Documents are assigned to categories based on these rankings. We demonstrate that precision and recall can be significantly improved by solving the categorization problem taking hierarchy into account. Specifically, we show that by adjusting the category levels in a principled way, that precision can be significantly improved, from 84% to 91%, on the much-studied Reuters-21578 corpus organized in a three-level hierarchy of categories.

1. Introduction and Background

The volume of online information has drastically increased with the explosive use of the Internet and online databases. Text retrieval systems employed by search engines for accessing this information have difficulty keeping pace with the growth in the amount of data that needs indexing and searching. Categorization of the original text is a method of organizing and making more efficient the retrieval task by sorting information into pre-specified “category bins” that can then be queried against using natural language processing systems.

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 adjusting the category level numbers.

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.

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 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. 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 weight 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 methods that used hierarchy obtained small increases in precision and large increases in speed [15]. As the number of categories increased in a corpus (OHSUMED), precision and recall decline to 60% [22].

In a previous paper [4] we show that it is possible to obtain more significant improvements in precision and recall by making use of the hierarchy. We describe an earlier version of the algorithm discussed here and show that treating the categorization problem within the context of a hierarchy is effective in realizing these improvements. The principal focus there was on the effect of the hierarchy itself and in refining the hierarchy. In some cases, moving categories from one place within the hierarchy to another within it can further improve the accuracy of the categorization. Here we extend that investigation and focus on the effect of adjusting the category levels to further

improve accuracy. We are particularly interested in exploring the situations where one approach (hierarchy modification or level modification) works best.

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 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.

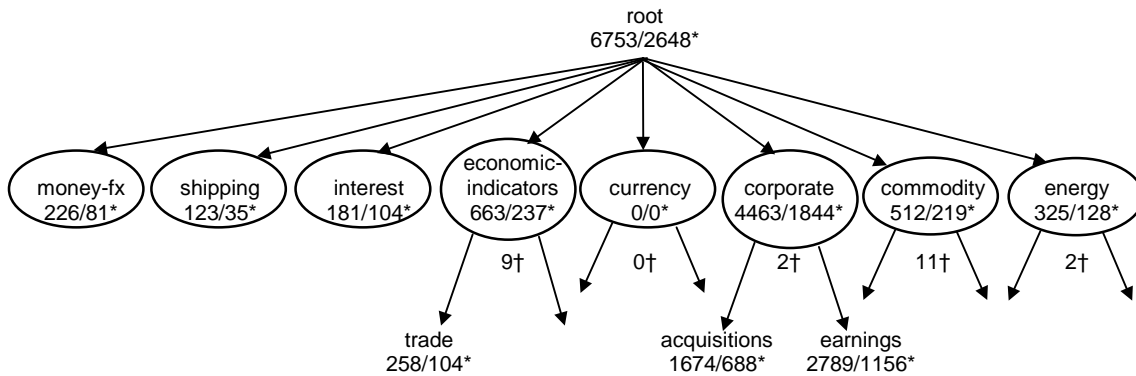
2.2. Document Corpus & Taxonomy

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-21578 collection uses 135 topics 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). The number of categories per supercategory varies widely from a minimum of 2 to a maximum of 78. All of the documents in the Reuters collection are assigned to 0 or more of the original 135 topics categories. In this case, documents are assigned only to leaf categories of the hierarchy while, in general, this is not necessarily the case.

The number of training documents per category also varies widely, from a minimum of 0 (for 71 such categories) to a maximum of 2,789 (earnings). On the other hand, document size does not vary greatly across categories. In the experiments described in this paper, we only considered categorizing test documents into categories having 20 or more training

documents. This was done in order to focus on a problem where there was enough statistical significance in the features we extracted to make comparisons among different category levels meaningful. This limited the investigation to 27 categories and actually removed only 94 documents (less than 3.5%) from the test corpus. This increased the overall precision and recall by about 1.5%. However, since we are principally interested here in studying the effect of varying the category level numbers, this is not a problem as all the experiments described were carried out on the same corpus.



* number of training/ test documents
 † number of subcategories in test set

Figure 1 Reuters basic hierarchy

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, respectively. 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 91% precision and 90% 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 two aspects that make our approach novel. 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, both leaf and interior, retaining the most significant words. This process is described in the next section.

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 child loses 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.

3.3 Assignment of Category Level Values

The focus of the experiments described in Section 4 is to investigate the effect of modifying the category levels in the ACTION Algorithm and in the ranking process which actually selects document categories. We begin with the root at level 1 and with all other categories at a level one higher than that of their parents. We run a categorization and measure the resultant precision and recall for each category and for the corpus as a whole.

Next, we consider the effect of varying the level of the root, observing the effect on accuracy, and setting the level of the root (and all other categories, since their levels are set relative to that of the root) to the best value found. A simple, linear search is carried out at a fairly coarse scale (increments of .25). Experiments we carried out using a finer scale did not yield significantly better results and we thus limited all the experiments here to this stepsize of .25. Even with such a simple search, we obtained significant improvements in accuracy, over 7% overall. It is our intention in the future, after examining the effects of the interaction between hierarchy modifications and level modifications in more detail, to return to the issue of searching over a narrower grid. At this stage in the investigation, however, we felt that doing so would only obscure the main results.

Actually, the category level numbers serve two purposes in the algorithm: word selection and document ranking. First, during the ACTION Algorithm (see Equation 5), they affect the competition (between parents and children) for words. A parent at level L will compete successfully with a child at level $L+D$, removing a word from the child's wordlist, if the F_p/F_c , the frequencies of the word in the subtrees rooted at the parent and child, respectively, exceeds $(L+D)/L$. Thus, the difference in the level number of the parent and child directly affects how high the relative frequency of the word must be, in the child relative to the parent, in order for the child to retain the word. Making D smaller strengthens the parent with respect to the child. On the other hand, making L smaller while leaving D the same, weakens the parent with respect to the child. But altering L is fundamentally different from altering D as altering L also affects the strength of the parent with respect to its own parent.

Thus, in modifying level numbers we must consider this interaction. We do so simply by looking for categories where the precision and recall are very different and where the interaction with other categories is marked. At each step, we consider the performance of a node relative to its parent, strengthening or weakening it as appropriate to balancing the node's precision and recall, specifically, its ability to attract the correct documents to its subtree.

Changing a node's level number also affects the ranking process. Again, the higher the level number, the stronger the node. Now, however, the change in level number also affects a node's strength with respect to its siblings as siblings compete directly for documents reaching their parent. We deal simply with this problem too. By examining the dispersion matrix, we observe which categories in the group under a common parent are too strong, aggressively stealing documents from their siblings, and which are victims. We begin by adjusting the node most out of kilter, or several nodes that are all out of kilter in the same direction and are not directly competing with one another. In practice this was found to be effective; experiments with more complex modification procedures did not produce significantly better results.

Actually, it is possible to consider two different level numbers, one for word selection and another for document ranking. In fact, the motivations for modifying a node's level number for word selection and for document ranking coincide thus making it reasonable to consider making similar adjustments. We plan to return to this issue as part of a broader investigation of refinements to the overall algorithm, preferring to concentrate here on the simpler case. Even using this simple approach, however, we obtained significant gains.

4. Computational Experience

There are a number of ways that the performance of a hierarchical categorization system can be tuned. Here we describe experiments performed in order to understand the effects of adjusting the level numbers (weights) of the categories within the hierarchy.

The purpose of this research is to investigate the role of a hierarchical organization of categories on the text categorization task. In particular we are considering a tree of categories with each node in the tree assigned a level number. As described above, this level number is used in evaluating the significance of features during the feature selection process, and in weighting of document features during the categorization process. The experiments reported here were conducted to determine the impact of this level number on feature selection and categorization of documents.

We begin with a base line case. We use the topics hierarchy supplied with the Reuters-21578 corpus, and consider only the leaf categories. We add a root category to make a simple tree structure. We assign the root a level number of 0 and give the leaves a level of 1. With this organization of categories and level numbers the root is unable to remove any features from a node during the feature selection process. Therefore, it effectively becomes a set of nodes rather than a tree. When we apply our categorization algorithm to the set of test documents we achieve a precision of 83.6% and a recall of 83.5%. We refer to this case as Flat-0. Note that if no category gives a document a ranking above our threshold, currently set to zero, then the document remains unclassified. In the Flat-0 case there are 2 unclassified documents.

We modified the base case by giving the root a level of 1, and all leaves a level of 2. The root is now capable of extracting features from the leaves during the feature selection process. When we apply our categorization procedure to the same test data as above we achieve a precision of 90.6% and a recall of 87.2%. We refer to this case as Flat-1. In Flat-1 there are 99 unclassified documents, but the precision and recall are significantly improved.

With a level number of 1, the root aggressively removes features from the leaves. The result is that 97 more documents receive rankings below the threshold and remain unclassified in Flat-1 than in Flat-0. We hypothesize that if the root were less aggressive in removing features from the leaves, the leaves would retain better features, resulting in better recall and precision. On the other hand, if the root has too low a level number the root does not remove any features from the leaves, and as a result the leaves retain features that are noisy. We tested this hypothesis by assigning the root a level of .75 and the leaves levels of 1.75. We refer to this case as Flat-75. Applying our feature selection and categorization algorithms as above resulted in a precision of 91.2% and a recall of 89.2%. In this case 60 documents were unclassified but both precision and recall were improved when compared to Flat-1. The results of the experiments on the test data for the three Flat hierarchy cases are included in the summary Table 3.

These results support our hypothesis that the value of the level numbers affects the ability of the root to remove features. We conducted a further experiment to confirm this conclusion. Normally our program removes stopwords from the training and testing documents. Since we restrict the number of features at each node to 50, this insures that the retained features are useful. We modified our programs so that stopwords were not removed, then ran the feature selection and categorization processes. If our conclusions regarding the level numbers were correct, then using a level number of .75 should result in precision and recall approximately equal to the results described above for Flat-75. However if we run the program with a root level of 0 the precision and recall should deteriorate since the stopwords will impede performance. When we performed these experiments we achieved a precision of 90.7% and a recall of 88.9% with a root level of .75 and a precision and recall of 78.3% with a root level of 0. These results confirm that our feature selection algorithm together with appropriate level values significantly reduces noise and improves performance.

Our next objective was to determine if the level numbers could be tuned to improve performance in the case of a more elaborate hierarchy. For this set of experiments we also used the topics hierarchy provided with the Reuters-21578 corpus (Figure 1). This time we included the intermediate categories, corporate, commodities, economic indicators, energy and currency. We first established a base line for performance by assigning the root a level of .75 and increasing the level numbers by 1 at each lower level of the tree. We refer to this organization as Base-Hier. We ran our feature selection program using the training data, and our categorization program using the same test data as above. The result was a precision of 87.1% and a recall of 85.2%. This result is

reported in the summary Table 3. In order to tune the level numbers we repeated a process of first using the training data to select a set of features for each node, then categorizing the training data and using the results to select new level numbers, then repeating the feature selection/categorization process on the training data until we arrived at an appropriate set of level numbers. At that point we could judge the effectiveness of the training by comparing our results against the base line case.

We began the process by using the training data for feature selection and categorization with Base-Hier. Based on our analysis of the results from the previous experiments we hypothesize that we could improve the categorization performance in two ways. First, if a category is achieving high precision and low recall, we could raise its level number; and second, if a category is achieving high recall and low precision we could lower its level number. For our first experiments we selected simple cases of nodes that were experiencing poor performance, and as we learned more about the tuning process moved onto more involved cases.

When we apply our feature selection and categorization programs to our training data using Base-Hier we get a precision of 89.2% and a recall of 87.5%. When we examine the results more closely we see that the categories of interest and money-fx are candidates for tuning. Interest has a precision of 95% but a recall of only 23% while money-fx has a precision of 89% and a recall of 60%. Both of these categories are direct descendents of the root and have no descendents. In both cases raising the level numbers should allow us to improve recall. We changed both level numbers from 1.75 to 2.75 and ran our feature selection and categorization procedures with the new hierarchy. Overall the precision and recall improved to 90.8% and 89.3% respectively. Interest has a precision of 98% and recall of 62% and money-fx has a precision of 85% and recall of 84%. Of course these results are from categorizing the training data, however they do indicate significant improvement.

If we look at the results of the previous experiment we see that with a precision of approximately 90% and a training set of 6493 documents, we are making approximately 650 errors. The largest single source of these errors occurs in the corporate subtree. Corporate has two subcategories, earnings and acquisitions. Earnings has a precision of 91% and a recall of 99% while acquisitions has a precision of 94% and a recall of 84%. The corporate category has a precision of 97% and a recall of 98%. These categories account for approximately 2/3 of the training data. From these results we can see that almost all of the earnings and acquisitions documents are correctly placed in the corporate category. Our categorizer must then decide if the documents are earnings or acquisitions documents. Our program is placing 22 earnings documents in the acquisitions category and 211 acquisitions documents in the earnings category. In addition, 42 earnings and acquisitions documents are left in the corporate category since there was no positive rank. In all, this is a total of 275 mistakes, which accounts for a substantial portion of the total 650 mistakes. Clearly this set of categories is a good candidate for tuning. This case is more complex than the interest/money-fx case since earnings and acquisitions are not descendents of the root. As we tune their level values we want to improve the

performance of earnings and acquisitions without having a negative impact on the performance of the corporate category.

Since many more acquisitions documents are being classified as earnings documents than the reverse and acquisitions' recall is significantly lower than its precision (see Table 1), we should lower the level number of earnings relative to acquisitions in order to make acquisitions stronger. At this point corporate is at level 1.75, and both earnings and acquisitions are at level 2.75. There are a number of ways that we might tune these levels, we explored three possibilities. The first alternative leaves corporate at 1.75 and acquisitions at level 2.75 but lowers earnings to 2.5. Call this A1. The second alternative leaves corporate at 1.75 and lowers both earnings and acquisitions, earnings to 2.25 and acquisitions to 2.5. Call this A2. The final possibility lowers corporate to 1.5 and earnings to 2.5 and leaves acquisitions at its 2.75 level. Call this A3.

We would expect that using A1 would result in acquisitions getting better features and consequently also getting more of its own documents, with the possible side effect of having more earnings documents classified as acquisitions. A2 makes corporate stronger relative to both earnings and acquisitions. As we saw in the flat cases this would mean that corporate would remove features more aggressively. We would expect therefore that acquisitions would have fewer of its documents classified as earnings, but there is the possibility that many more documents from both earnings and acquisitions will be unclassified. A3 makes both earnings and acquisitions stronger relative to corporate and we would expect to have fewer unclassified documents. We would also expect that fewer acquisitions documents would be classified as earnings. Since both A1 and A2 leave corporate at level 1.75 we would also expect that corporate would continue to achieve both high recall and precision. In fact, other branches of the hierarchy should be unaffected by the changes inside the corporate tree (one of the strengths of a hierarchical approach). A3 changes the level of corporate so there is the possibility that the performance of corporate relative to the rest of the hierarchy will deteriorate in this case.

We ran our feature selection and classification procedure for all three cases. The results are shown in the Table 1 below.

	Level	#s		Overall Prec/Rec	Earn Prec/Rec	Acq Prec/Rec	Unclass Corp Docs	Earn as Acq	Acq as Earn
	Corp	Acq	Earn						
Before	1.75	2.75	2.75	91/89	91/99	94/84	42	22	211
A1	1.75	2.75	2.50	93/91	96/98	92/93	55	47	55
A2	1.75	2.50	2.25	92/90	97/97	91/91	96	65	39
A3	1.50	2.75	2.50	91/89	97/94	88/93	15	160	50

Table 1: Precision, Recall, Unclassified Corporate Documents, Earnings Documents Classified as Acquisitions and Acquisitions Documents Classified as Earnings for Different Levels Number for Corporate, Acquisitions and Earnings using Training Data.

As we can see A1 allows acquisitions to retain better features and its recall improves significantly. By making earnings weaker it classifies fewer acquisitions documents as earnings and earnings achieves a higher precision with a slight decrease in recall. The weaker value for earnings also results in more unclassified corporate documents. A2 produces similar improvements in the precision of earnings and the recall of acquisitions but results in many more unclassified corporate documents resulting in a slightly lower overall recall. A3 gives the fewest unclassified corporate documents. Since corporate has a lower level number in this case it does not remove features as aggressively as in the other two cases. One side effect however is that many more documents are incorrectly classified as acquisitions, and the overall performance deteriorates. Of course there are other adjustments that could be made, but our objective was not to find the optimum combination, but rather to understand the effects of changing the levels. We selected A1 as the best alternative.

The next case we consider is the economic indicators subtree. This is a more complex case than those described above. Nine of the categories in this subtree have more than twenty training documents and are used in these experiments. Together there are 663 training documents for the categories in the subtree. Using the A1 hierarchy above the subtree achieves a recall of 88% and a precision of 84% on the training data. Within the subtree the performance is quite varied. Five of the nine subcategories have a precision of over 90% while four of the categories have recall below 70%. In some cases the difference between precision and recall is very large. The category *cpi* for example has a precision of 100% but a recall of only 40%. Balance of payments has a recall of 88% and a precision of only 29%. On the other hand, *trade* has a precision of only 73% and a recall of 88%.

All the categories within economic indicators have level 2.75. We tested our hypotheses regarding the effects of level numbers by adjusting the levels within the subtree. We increased the level of the two nodes with very low recall and high precision from 2.75 to 4.75. We increased the level of one node to 3.75 and we decreased the level of *trade* from 2.75 to 2.5. Nodes with recall and precision approximately equal were left unchanged. With these adjustments, our overall performance on the training data was a 93.0% precision and a 91.7% recall. Using our guidelines we performed a final round of tuning throughout the hierarchy (called Final-Hier) using the training data with a precision of 93.2% and a recall of 92.0%. The results of these experiments on the training data are reported in Table 2.

We then used the resulting hierarchy, Final-Hier, to categorize the test data. The result was an overall precision of 91.5% and a recall of 89.9%. This compares favorably with our results on the test data using Base-Hier where we achieved a precision of 87.1% and a recall of 85.2%. See Table 3 for a summary of selected corresponding results using the test data.

We performed additional experiments to test the robustness of our final hierarchy. In all of the experiments above we restricted ourselves to categories that had at least 20 training

documents. In the first test of robustness we relaxed this condition and only required 10 training documents. When we applied our categorizer to the test data we achieved a precision of 91.0% and a recall of 89.4%. In our second test we relaxed the condition further and considered all the categories regardless of the number of training documents. When we applied the categorizer in this case we achieved a precision of 90.0% and a recall of 88.4%. In our next test we kept the level values of the categories the same but retrained the graph using only 30% of the data as training data. We then tested the categorizer on the remaining 70%. In this experiment we again required 20 training documents for a category. The result was a precision of 89.8% and a recall of 89.4%. Finally, we tested the categorizer on an alternate 70/30 random split of the corpus and obtained similar results. This final result is also reported in Table 3.

	Precision(%)	Recall(%)
Base-Hier	89.2	87.5
Interest/Money-fx at 2.75	90.8	89.3
Earnings at 2.5 (A1)	92.7	91.0
Adjusting Economic Indicators	93.0	91.7
Final-Hier	93.2	92.0

Table 2: Results Using Training Data

	Precision(%)	Recall(%)
Flat-0	83.6	83.5
Flat-75	91.2	89.2
Flat-1	90.6	87.2
Base-Hier	87.1	85.2
Final-Hier	91.5	89.9
Final-Hier (Alt split)	91.1	89.9

Table 3: Results Using Test Data

5. Summary and Conclusions

In this paper, we have explored the effect of modifying the category level numbers in an algorithm for hierarchical text categorization and have shown that it is possible to obtain substantial improvements in precision and recall by doing so. Specifically, we improved precision and recall from an 84% level to over a 91% level, by adjusting the category level numbers. The procedure we used was a simply, greedy search heuristic guided by the principle that categories whose precisions significantly exceeded their recall were too weak and those whose recall exceeded their precision were too strong.

In a previous paper [4] we explored the effect of modifying the hierarchy itself, moving categories from one part of the hierarchy to another, in order to achieve similar objectives. We found that approach effective also and have now shed additional light on the role of hierarchy in the categorization process and in the interaction between hierarchy modification and level modification. Close examination of the dispersion matrix has been

very useful in this regard. We found that level modification was most useful in cases where a category was generally too weak or generally too strong. The row or column in the dispersion matrix containing many off-diagonal elements characterized these cases. On the other hand, when the problem was a single large off-diagonal element, moving a category from one part of the hierarchy to another was more effective. In some cases, both approaches were effective.

We have seen examples of all these cases. We illustrated we could achieve improvements by modifying the level numbers for earnings and acquisitions or, alternatively (in our previous work [4]) by altering the hierarchy by removing the intermediate corporate category. The former approach, however, worked somewhat better. We found that we could gain by altering the level numbers of interest and money-fx or, alternatively making them children of economic indicators. Both approaches worked, but in this case, the latter worked better.

Based on our computational experience to date, our conclusion is that both types of adjustment are useful and that much of the obtainable gain can be achieved by making adjustments individually, focussing on simple adjustments and on those with large potential gains. Our next goal is to explore this interaction more closely and to automate the process of category level number modification. We also plan to explore the use of these techniques in problems with multi-category documents.

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. Hierarchical Text Categorization. Technical Report XXXXXXXX (URL).
- [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-Kauffman, 1997.
- [25] UMLS Knowledge Sources 8th Edition; National Library of Medicine, January, 1997.